Internet with Transient Destination-Controlled Addressing

Zakaria Al-Qudah, Eamon Johnson, Michael Rabinovich, and Oliver Spatscheck

***Abstract*—Today's Internet makes hosts and individual networks inherently insecure because permanent addresses turn destinations into permanent attack targets. This paper describes an *Evasive Internet Protocol (EIP)*, a change to the data plane of the Internet that (a) prevents senders from forging their identities while preserving the current Internet privacy paradigm, (b) gives recipients full control over who can communicate with them and for how long, (c) achieves the above features without requiring global signaling protocols, and (d) allows coexistence with and graceful introduction into the current Internet. We motivate our approach, present the architectural design, and evaluate it through trace-driven and synthetic simulations as well as prototype testing.**

Keywords*—*Internet protocol, Internet architecture, network security.

I. Introduction

As the Internet permeates through every aspect of our lives, the stakes are rising in ensuring its continued viability. The Internet is assaulted from multiple fronts. Spam has already changed the social norms of using email, reflecting a new reality that legitimate mail may never be read due to being entangled in spam filters. Malware dogs peer-to-peer networks and open source software distribution. Denial of service attacks against network infrastructures and Web sites have become routine. Computer break-ins and hijacking are wide-spread. Identity theft through phishing or break-ins is on the rise. The importance of these issues has been well recognized and is one of the main reasons behind the recent push by the networking community to re-engineer the Internet [1].

A fundamental challenge in designing a more secure Internet is to satisfy the competing demands of security, openness, and user privacy. Indeed, preventing bad actors in an open environment seems to entail being able to hold actors *accountable* for their actions, which in turn suggests being able to *attribute* the actions to particular users, which undermines user privacy. The goal of the proposed approach is to address this challenge, and to do so not by selecting a particular tradeoff point in this tussle space but by providing the tools that would give users the flexibility to select their own tradeoffs between openness and security, while at the same time not affect the privacy paradigm of the overall network architecture within which our approach is used (e.g., the current Internet).

Al-Qudah is with Yarmouk University, Jordan. He did this work while visiting Case Western Reserve University. Johnson and Rabinovich are with Case Western Reserve University. Spatscheck is with AT&T Labs – Research Contact author: M. Rabinovich, michael.rabinovich@case.edu.

This work was supported in part by NSF under Grants CNS-0831821 and CNS-0916407.

By design, our approach is not a new overarching architecture for the Internet. Rather, we strive for a *minimal* change in the current Internet that would (a) prevent senders from forging their identities while preserving the current privacy paradigm (which is useful in itself but, more importantly, enables the next feature), (b) give recipients full control over who can communicate with them and for how long, and (c) achieve the above features without requiring global signaling protocols. We argue that these capabilities would eliminate a wide range of attack vectors and provide a foundation for addressing other attacks at a higher level.

Our approach is based on a general notion of network *capability* [2], [3], which represents authorization to communicate with a host, and which is valid only for a specific sender and for a limited amount of time. While capabilities are well known (although our flavor of the capability is different) our approach incorporates three key new ideas. First, capabilities are the *only* means for hosts to be reached; in particular, hosts do not handle capability requests and thus are not vulnerable to the *denial of capability* attack [4]. This removes a major obstacle in capability systems, which had to employ complex router mechanisms such as per-path hierarchical fair queuing in TVA [5] (which as we show in Section VII-C may still fall short of providing enough protection) or proof-of-work in Portcullis [6] and Speak-up [7] (which can delay communication and drain batteries in user devices). Second, capabilities are distributed by the name system (e.g., DNS) rather than recipient hosts or distinct infrastructure. The DNS servers are themselves protected through capabilities, except for the root servers, which are widely replicated and protected by anycast addresses. Finally, a capability in our approach can be independently constructed by the destination network and authenticated by any point in the network, in contrast to existing approaches where the routers on the path between the sender and destination contribute *pre-capabilities* to capability construction. Although regular IP addresses are still used for routing, hosts in our approach can only be reached through capabilities, which are *transient* due to their limited validity. We named our approach Evasive Internet Protocol (EIP) to reflect its lack of permanent means of reaching a host.

While we have paid careful attention to factoring EIP into today's Internet, we hope it will also prove useful to the efforts in re-architecting the Internet currently underway in both the US and Europe [1], [8]. As an approach to enhancing network-layer security, EIP explores a point in the design space that should inform these ongoing efforts.

We presented preliminary ideas behind this work in a position paper [9]. The present work fleshes out and evaluates these ideas.

II. Vision

Given the tussle space between security, openness, and privacy, our goal is to empower the Internet end-points (the end-hosts and their access networks) to impose their own policies in this regard. At the extreme, a host can allow incoming traffic only from a known set of destinations, and EIP will prevent other destinations from forging their IP addresses to bypass this policy. Short of this extreme, the recipient can dynamically adjust traffic constraints imposed on various external destinations based on their prior behavior, the recipient's current load, or any other factors as deemed appropriate. Furthermore, we strive for a minimal change in the current Internet architecture that would allow this empowerment. Our vision rests on the following main components.

T-address We envision Internet destinations to be represented by two distinct concepts – IP addresses and capabilities. The IP addresses would still be used in routing algorithms for route computation and forwarding table indexing. This allows us 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 [10], [11], [12].

Although IP addresses are used for route computation and forwarding tables, the host IP address in our architecture cannot be used to communicate with the host: a compliant router will only forward a packet with a valid destination capability. Thus, from the perspective of senders, the capability in effect becomes a transient destination address – referred to as a *t-address* below. Furthermore, a capability can only be issued by an authorized entity, such as the name server acting on behalf of the host, and the chain of authority is cryptographically enforced, as are the credentials of the sender. The authorization enforcement prevents a sender from tricking routers to forward a packet with a forged capability.

Incremental deployment We design the t-address to include all the information needed to verify its validity, which can thus be done at any point in the network, and in the worst case by the recipient itself. Further, we envision EIP to be (at least initially) implemented as a shim over the current Internet. Together, these aspects will allow EIP to coexist with the current Internet when incrementally deployed. In fact, two EIP subnets can communicate over EIP even if they are connected entirely through the legacy Internet; EIP behavior in this scenario reduces to having two "firewalls on steroids" in front of the subnets.

Preservation of privacy paradigm. Because EIP uses IP addresses to identify communicating parties, it in itself does not undermine user privacy – the privacy is as preserved (or violated) as in today's Internet. All EIP does is remove the ability for a sender to lie about its IP address and empower a recipient to control how much traffic it is willing to receive from which IP addresses and for how long. A consequence of this design decision is that any such traffic policies apply at the granularity of IP addresses, which may not correspond to individual hosts as some hosts can be multi-homed or fronted by a network address translating (NAT) device. Again, this is similar to today's Internet where blacklists are commonly keyed by IP addresses except EIP provides a systematic framework to support these, as well as richer, policies. If our vision is incorporated into future Internet architectures with stronger notions of identities, then of course the privacy implications of these identities would carry over. But we leave these design decisions outside EIP itself.

Assumptions EIP assumes support from ICANN (to bootstrap certificates attesting the ownership of IP addresses) and DNS roots (to offer EIP protection to lower-level authoritative DNS servers), which existing capability systems do not require. While ICANN already provides the needed support through the RPKI mechanism [13], support from DNS roots would require their buy-in. Given that EIP eliminates the need for costly mechanisms to protect against the denial of capability attacks, we believe this represents a beneficial tradeoff and makes EIP a realistic direction for adopting network capabilities into the Internet architecture.

To benefit from EIP, a destination network would need to implement changes to its DNS servers (the authoritative servers in the case of Web sites and resolvers in the case of client access networks) and either externally accessible hosts or an appliance (a firewall, a router, or a switch) in front of the hosts. The host modification for EIP support is limited to the IP layer of the kernel's network stack. Modern network appliances, such as those with OpenFlow support, offer an interesting alternative (to modification in end-hosts) for EIP implementation which we leave for future work.

Finally, routers throughout the Internet can optionally be modified to support EIP, which allows them to independently police the traffic flowing through them. This involves extra logic to verify incoming datagrams and maintain the flow records cache. The rest of the router functionality, including the routing protocols and forwarding logic, remains unchanged. EIP support is especially useful in edge routers that connect destination networks to the Internet, and this paper assumes that, while legacy networks may connect to legacy routers, an EIP subnet is connected to the Internet through an EIP-compliant router.

III. Significance

To highlight the benefits of our vision, we show how it helps defend against some specific high-profile attacks that dog today's Internet.

Spoofing Attacks In spoofing attacks, an attacker injects into the network packets with forged source IP addresses. Currently deployed[1] anti-spoofing defenses rely mostly on ingress address filtering [14] and unicast reverse path forwarding [15] but their effectiveness is reduced by multihoming and asymmetric routes [16]. Thus, spoofing-based attacks continue to occur and exert damage. These attacks range from old SYN-flood attacks [17] to DNS poison attacks using spoofed DNS responses [18] to reflected denial of service attacks with spoofed DNS requests [19], to TCP session disruption through injected spoofed TCP reset segments [20]. Specific mechanisms have

[1] See Section IX for research approaches.

been proposed to counter some of these of these attacks, e.g., TCP SYN cookies to protect against SYN floods, or DNSSec against DNS poison attack. EIP will make spoofing attacks organically impossible: since a t-address of the destination is source-specific, any packet with a spoofed source IP address will be dropped by the first EIP-compliant router.

Multistep (Fast-Flux) Attacks Multistep attacks use botnets to evade detection. These attacks employ so-called fast-flux networks [21] of compromised hosts used as front-end Web servers. In the simplest form, the attacker directs clients to these Web servers by using a public DNS service to map the hostname of a spam URL to a rotating set of compromised hosts' IP addresses. The front-end Web servers then communicate with the users and act as reverse proxies for a back-end fraudulent Web site, so that the latter is hidden from detection. These attacks have become widely spread, e.g., Holtz et al. [22] found that almost 30% of all domains used in spam URLs are hosted by fast-flux networks. Clark and Landau consider multistep attacks "the most challenging set of attacks to investigate and deter" [23].

Much effort has been spent in trying to detect fast-flux networks (e.g., [22], [24]). By enforcing a requirement that authoritative DNS servers only map host names to IP addresses with the authorization from the hosts' network providers, EIP makes any fast-flux-hosted attacks impossible.

Denial of Service Attacks Denial of service (DoS) attacks, including distributed DoS (DDoS) attacks, on the Internet have unfortunately become commonplace. These attacks differ in sophistication – some are harder to detect than others. The big problem, however, is that they are often hard to block even after detection. Indeed, even if fraudulent packets are dropped by an ingress firewall at a site, they still consume bandwidth of the site's connection to the Internet. And if these packets are dropped by the end-host (typically server), they further consume server resources for receiving and recognizing them.

EIP equips system administrators with necessary tools to counter DoS attacks. Attack packets with fake source addresses become impossible as discussed earlier. Packets with genuine source addresses can be blocked by the destination *long before* they reach the destination network. This is true even for sophisticated attacks where a botnet targets different hosts within the destination network. Indeed, security alerts against malicious senders can be made available to both the authoritative DNS server and the interested hosts within the site, so that they can deny or not renew capabilities to these senders.

Application-Level Attacks As a network architecture, EIP by definition cannot in itself eliminate application-level attacks. No network architecture can prevent a user from submitting their private data to a phishing site, or clicking on a fraudulent URL and downloading malware, or installing a trojan along with some free software. EIP does not supplant research into hardening hosts against viruses and break-ins. Rather, it provides a foundation for addressing these attacks at a higher level. For example, consider a Web server under a password-guessing attack at the HTTP level. Network-layer mechanisms are unlikely to detect it but the HTTP server application can. With EIP, the Web server can stop renewing the capabilities for the offending host(s) and instruct its authoritative DNS server to do the same. This is more efficient than discarding these packets at the host's or ingress firewall, which must match *every* incoming packet against its filtering rules. Similar considerations apply to other application-level services such as email and ftp servers.

Application-level attacks are also frequently employed as a denial of service weapon. For example, an attacker may bombard a Web service with well-formed HTTP requests to a devastating effect [25]. Current proposals to protect against this threat are mostly concentrated within the Web server itself [26], [27] or through a layer of shielding nodes in an overlay network [28], [29]. EIP enables fine-grained capability-issuing policies that can help the site to protect against this threat at the network level. If the attacking hosts are identified, the site can stop their flows before they hit the site network Otherwise, the site can employ fine-grained traffic control. For example, new senders might initially receive authorization to send only a small amount of traffic, with subsequent authorizations becoming more generous depending on the sender's behavior and the overal load at the site. Basically, the goal is to apportion authorizations among senders in a way that would not saturate the site's resources while factoring in senders' past behavior in making these decisions. Investigating appropriate policies is outside the scope of this paper, but the EIP architecture provides the necessary *mechanism* to enact them.

IV. The Architecture

We now sketch the main architectural components behind EIP. This section presents an ideal design assuming jumbo frames[2] In practice, EIP can fit into the currently prevalent MTU of 1500 bytes as shown in Section VI.

A. Delegation of Authority

Any entity issuing a destination's t-address to another party must provide evidence that the issuer is authorized to issue the address. Otherwise, a rogue entity can issue a t-address to a destination without the destination's consent, thus taking control from the destination over incoming traffic. Similarly, an entity requesting a t-address must provide proper credentials to the issuer, to allow the issuer to make a reasoned policy decision on granting the request, and also to grant the permission to the response to reach the requester (indeed, the response represents incoming traffic for the requester).

Leveraging the RPKI framework [13] that is already being deployed on the Internet, we assume that whenever an entity obtains a block of IP addresses from a provider, the entity also obtains from the provider a certificate chain verifying the right to use this block. The certificate chain reflects the network provider hierarchy and emanates from ICANN. While RPKI hierarchy typically stops at ISPs, it allows a provider to further issue self-published certificates to its customers—the feature

[2]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 frames is relatively low bandwidth: assuming Cisco's rule of thumb of keeping frame transmission within 10ms, roughly 10Mbps is needed to serialize a jumbo frame within this limit. Access networks are moving beyond this limit in both uplink and downlink speeds.

leveraged in EIP. These certificates are small – under 300 bytes (256 bytes for the signature and the rest for the IP block, the hash of the recipient's public key, and the key expiration time). Furthermore, an end-host in EIP also obtains its certificate along with its IP address, presumably through DHCP.[3] Then, the issuer of a t-address can attest to its authority by including appropriate certificates as discussed later.

There are two types of entities that can issue a t-address for a destination to another party: the authoritative name servers and the destination hosts themselves.

Name servers. We envision that authoritative name servers (ADNS) will be the primary entry points to the destinations whose names they maintain: these name servers will be issuing the initial t-addresses to these destinations in response to the DNS queries for the destinations. Even when a host already knows the destination's IP address (as, e.g., in BitTorrent), the host must still obtain the destination's t-address using a reverse DNS query before initiating communication. To issue t-addresses, as part of the DNS service setup, for each block of IP addresses the name server will be handling, it must obtain a certificate from the corresponding ISP, which signs the IP address block. In most cases, the authoritative name server will belong to the same ISP as the destinations whose names it is resolving. In other cases, e.g., in some content delivery networks such as Akamai or when a domain name is handled by an external DNS service, the authoritative DNS server will map a name to destinations in different autonomous systems; in these cases, the name server would obtain a separate certificate for each address block from its corresponding provider.

End-hosts. A host can always issue its own t-address to another party. Because the end-host can only be reached through a valid t-address to begin with, this ability is utilized only in two scenarios. First, when the host initiates communication with another party (i.e., acts as a client), it issues its source t-address so that the other party will be able to respond. Second, to ensure continued communication, the host can issue its new t-address to the other party during an ongoing communication before the old t-address expires (a *continuation t-address*).

Authority revocation. A practical deployment of our approach would require a mechanism for key revocation. Instead of developing a separate control protocol, we rely on implicit revocation through certificate expiration. Each certificate authorizing a key-pair includes the issuance timestamp (expressed as 8 byte absolute UNIX time) and the time-to-live (two-byte number of seconds) after which the certificate must be reacquired by the recipient.

B. Addressing

As mentioned before, the sole means of reaching a destination is through its t-address. The t-address contains the destination IP address (16 bytes – enough for IPv6 addresses), the validity constraint (4 bytes), and the issuance timestamp (9 bytes – 8 bytes for the UNIX timestamp in seconds plus an extra byte for finer granularity time ticks, allowing 4ms increments, which is less than prevalent wide area network delays). The timestamp provides the basis for the time validity of this t-address (see below) and also allows the discarding, with high probability, of replayed packets, that might be issued in man-in-the-middle attacks. The t-address binds this information with the sender, by including either sender's IP address or, as discussed in Sec. IV-D, a 16-byte hash of sender's *client group ID* and public key. To ensure the uniqueness of t-address in the presence of duplicate DNS queries or multiple senders sharing a common IP address behind a NAT box, the t-address includes a 2-byte counter that either counts the number of t-addresses generated within a time-tick (if this t-address was generated by an ADNS in response to a DNS query, the counter size being sufficient for 16M queries per second) or simply provides a unique label for the sender (if this t-address is generated by the receiver as a continuation t-address, the counter size allowing to enumerate 64K concurrent clients - the maximum that can share a NAT). The t-address also includes a signature over the above information (256 bytes), the corresponding public key needed to verify the signature (256 bytes) and a certificate attesting to the authority of the issuer to issue the t-address. The latter signs the issuer's public key and the IP range with the destination provider's private key and includes the hash of the provider's public key. We assume that the providers' public keys and certificates are widely cached, or can be obtained as needed from RPKI repositories [13]. This hash can be small (4 bytes) because it is only used to locate the corresponding certificate in the cache and not for any crypto operations.

The validity constraint includes one byte for a timing constraint and three bytes for the bytes constraint. The former tells for how many seconds since the issuance timestamp this t-address remains valid. The latter represents the maximum amount of traffic in KB that can be sent in this time. This leaves us with a 846-byte t-address, resulting in datagrams that would not fit into MTUs of many of today's networks. We discuss this and a related issue of header overhead below in Section IV-C and VI.

C. Datagrams

EIP assumes loosely synchronized clocks among hosts and routers on the Internet. A host can obtain an initial time reading from a number of sources, such as a GPS receiver (especially for mobile devices where these receivers are now ubiquitous), a periodic time signal broadcast on a LAN (especially for institutional LANs where these signals are common), from DHCP, from a layer-2 tunnel to an NTP server, or through manual setup. Having obtained the initial time, the hosts can maintain it using either the above mechanisms or regular NTP.

Our current design calls for two types of datagrams as shown in Figure 1, with several format variations in each type (all distinguished by different values of the Type field). 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

[3]Note that our approach is not affected by well-known DHCP vulnerabilities even if they are not addressed through relay agents [30] or DHCP authentication [31]. Indeed, cryptographic properties of a certificate chain ensure it cannot be forged by a rogue DHCP server, and a simple use of self-generated public key by the hosts (sending the key to the DHCT server and having DHCP server return the certificate encrypted with this key) can ensure than a rogue host does not sniff a certificate chain from another host.

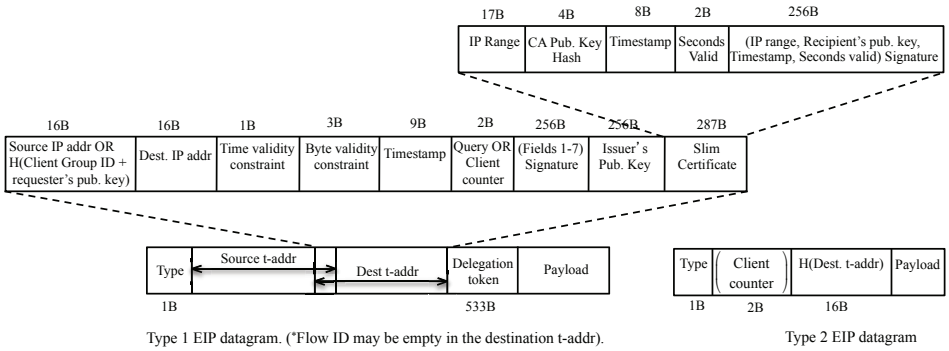


Fig. 1. Datagam structure.

and an optional *delegation* token, whose purpose will become clear in Sec. IV-D. To reduce the header size, we note that the destination IP address is embedded in both source and destination t-address; we include it only once, in a sense overlapping the two t-addresses in the EIP header.

Type-1 datagrams still have a large header – over two KB. We borrow the approach from TVA [5] to amortize the header overhead over subsequent communication by including only a flow ID in the subsequent packets, and we use a 16-byte hash of the destination t-address as the flow ID. In fact, our current design follows TVA in assuming these FIDs are sent unsigned for efficiency, relying on short lifetime of these FIDs for a "good enough" protection against fraudulent packets reusing the same FID. This design choice reflects our view on the current costs/risks landscape; we discuss it further in Sec. V.

Accordingly, the subsequent communication uses type-2 datagrams. When the corresponding destination t-address does not identify a flow from an individual sender, type-2 datagrams also include the client counter, which allows the destination to keep track of the amount of bytes transmitted by individual senders and issue further continuation t-address to the senders as needed. The address-related header size of type-2 datagram is only 17-19 bytes; Sec. VII-A shows that these datagrams will carry most of traffic.

Finally, before its t-address becomes invalid, a host may choose to include a *continuation* t-address with its next packet to the other party, if it is willing to continue the communication. This is done with an optional header field in type-2 datagrams, adding 846B (t-address size) to the header. To continue being able to track the volume of traffic coming from the other side, the host includes a unique (among sources with the same IP address holding an unexpired t-address for this host) client counter into its continuation t-address. The continuation 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.

D. Obtaining a Destination T-Address

To communicate with other hosts on the Internet, a host must first establish the ability to communicate with its local DNS server (LDNS), which in EIP means obtaining the LDNS's t-address. The host obtains it either from its DHCP along with the LDNS's IP address or, if the host wishes to use a third-party resolver (such as Google's Public DNS), through a reverse DNS query on the resolver's IP address[4]. As today, the host can also act as its own LDNS if it is willing to forgo the benefits of a shared DNS cache and if its ISP allows it.

Then, to communicate with another host, the source sends a DNS query with the destination hostname to the source's local name server (LDNS), which routes the query through name service infrastructure to the destination's authoritative DNS server. Because the destination t-address is bound to the sender, the LDNS would seem to have to forward the client's IP address along with the query. However, this would make the received t-address usable by this client only and prevent cross-client DNS response caching at LDNS. Consequently, the LDNS submits with its query a *client group ID*, which has the same format as a regular IP address but may or may not represent a real client[5] so that the received t-address will be bound to the group ID (and not the client's IP address).

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 embeds the source host's IP address) will no longer match the source IP address embedded into the destination t-address (which is the group ID). We bridge this gap with the help of a delegation token that LDNS returns to the host along with the destination t-address, and the host then includes into its type-1 datagrams. First, the delegation token binds the client IP address to group ID. Second, it apportions the byte constraint obtained for the whole client group among individual clients, by including a *byte sub-constraint* for the given client. The sub-constraints given to all the clients sharing this t-address must add up to no more than the byte contraint from the t-address (and a cheating LDNS that violates this rule will not undermine EIP security properties - see Sec. V). Third, to distinguish clients fronted by a NAT and thus sharing an IP address, the delegation token includes a *client counter* which is incremented for each query from a given IP address for a given domain name. Thus, the delegation token occupies 533 bytes: 2 bytes for the client counter (enough to enumerate 64K clients – the most that can

[4]In the latter case, the host would itself interact with the DNS infrastructure to resolve the reverse query; the host can do this because if its provider blocked direct DNS traffic the host would be unable to use an external resolver anyway.

[5]For IPv4, it could be an unroutable address, e.g., from the 10.0.0.0/8 block, and is thus unaffected by address shortage.

share a NAT), 3 bytes for the byte sub-constraint, 16 bytes for the group ID, 256 bytes for the signature (which signs the above fields plus the client IP address, which need not be included since it appears elsewhere in the packet), and 256 bytes for the LDNS's public key. Any router receiving a packet with this delegation token can verify that (a) the sender's source address was properly bound to the group ID by the owner of the public key from the delegation token, by verifying the delegation token signature, and (b) this public key owner is the same LDNS that obtained the t-address from the destination, by computing the hash of the group ID and public key from the delegation token and comparing the result with the corresponding hash from the t-address.

Once the source has the destination's t-address, it can send packets to the destination, and it may include its own t-address granting the destination a permission to communicate back. This t-address is bound to the destination's IP address directly and not hashed with the destination's public key (which is unknown to the sender and not needed since there is no delegation token to be verified). All continuation t-addresses are also bound directly to end-hosts' IP addresses as well as the client counters, which in this case appear in place of the query counter as shown in Fig. 1. Type-1 datagrams in this case carry no delegation token, and the subsequent type-2 datagrams have no client counter (since it is already embedded into the destination t-address).

This mechanism allows LDNS caching despite destination t-addresses being client-specific. However, the validity constraint in the destination t-address is now shared among all clients using the corresponding group ID. Thus, the destination's policy for validity constraint assignment becomes coarser grained – it applies to all clients in the group. While this issue deserves a separate investigation, we note that studies have shown that most benefits of shared DNS caching are achieved with small client populations [32], [33], with the cache shared by ten clients exhibiting virtually the same hit rate as the cache shared by thousands of clients. This limits the policy granularity coarsening effect to small client groups. Furthermore, the LDNS has its discretion in choosing which clients to put into the same group ID, e.g., grouping clients on the same subnet. In this way, the clients that share validity constraints may in fact be equally affected by the presence of a malicious host on the subnet and should be treated similarly (and often are, being quarantined as a group by sysadmins).

ADNS servers are in turn protected by their own t-addresses, which are given by the higher-level DNS servers in the DNS hierarchy. We expect that these t-addresses would have long TTL (to allow caching of NS records) but low byte limits. This limits the amount of load each querier can impose using one t-address, and new t-addresses can be denied by the higher-level DNS server to a misbehaving querier.

In this arrangement, each DNS server is protected by a higher-level DNS server in the DNS hierarchy. This leaves root name servers exposed since they must be well-known and permanently reachable just as in today's Internet[6]. Some sort of residual vulnerability is inevitable in any system for bootstrapping security. However, the scope of this vulnerability in our architecture is limited to just a few well-administered and widely replicated 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. (See [34] for a recent analysis of root resiliency to DoS attacks.) Thus, we believe narrowing the scope of vulnerability to the root DNS servers is a sensible design choice.

[6]A simple way to achieve the universal reachability of roots is to make their "private" keys well-known, allowing any host to mint a valid root t-address.

E. Routing

As mentioned earlier, EIP routers inherit the current routing protocols such as BGP for route computation. However, packet forwarding changes: no EIP router will forward a packet addressed with an IP address – the valid t-address is required.

When processing a type-1 datagram, a router verifies the validity of the destination t-address and that the source IP address from the source t-address matches the one from the destination t-address (using the delegation token if present as a bridge). The router forwards the packet that passes these checks according to the routing entry for the destination's IP address, otherwise it drops the packet. For a valid packet, the router creates a cache entry for its destination t-address indexed by the hash of the destination t-address – unless the entry for this t-address already exists in which case the router increments byte count in the existing entry. The router can follow the elegant mechanism from [5] to keep up with the line speed with a fixed cache size (see [5] for details), although our evaluation in Sec. VII-B indicates this may not be needed in practice as the router state is manageable regardless.

Type-2 datagrams are only processed if there is a cached entry for this flow – otherwise the router drops the datagram. In particular, any route change during an ongoing communication will result in a dropped packet, which would presumably be retransmitted by the end-host using type-1 datagram. However, beyond routing pathologies and intra-network routing, routes are generally stable in practice [35], [36].

The processing of type-2 datagram types involves simply updating the byte count for the flow and forwarding the packet. While processing type-1, most traffic will be carried by type-2 datagrams, which require trivial processing. Furthermore, if two peering EIP routers trust each other, the downstream router may choose to blindly forward packets arriving on this peering link without validating them, assuming that the upstream peer already has done so.

Note that the router creates a flow entry keyed to the information provided exclusively by the flow destination. Thus, a malicious sender cannot manipulate its packets to create spurious router state. 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.

F. Scalability Implications

EIP's reliance on asymmetric keys brings a potential concern about router scalability, especially that an attacker may attempt

to overwhelm a router with malformed type-1 datagrams, with the goal of forcing the router to perform a large number of signature verifications. However, modern security chips make our approach feasible. For example a Nitrox adapter CNN35XX-NHB4-G [37] is capable of 1 million RSA-1024 ops/sec. This translates into 4-6Gbps of type-1 EIP datagrams per card – enough to sustain a significant attack (and certainly more than plenty for legitimate traffic as only a small fraction of packets are verified as Sec. VII-A indicates). Further, since the attack traffic will be dropped at the first router, the rest of the routing fabric will not be affected, complicating an attempt to target a router from a large botnet.

Another potential concern is the load on DNS servers. First, communication using bare IP addresses, which does not require a DNS lookup today, would now need to generate a DNS query to obtain a t-address from the appropriate ADNS. We believe these extra queries are an acceptable cost for the EIP benefits. A more significant issue is the burden of maintaining the policy rules for assigning validity constraints to various senders on the authoritative DNS servers. While details on how to organize this state are left for future work, our straw-man solution is based on the following idea. The policies would be defined in terms of a certain number of "levels of service", one of which being a default. Each level will have a corresponding counting Bloom filter providing a compact representation of all the external requesters assigned to the given policy level. When processing a DNS query, the server will apply the highest policy level whose filter contains the requester's IP address (thus not penalizing the requester for an occasional false positive in the Bloom filters) – or the default level. If needed, the number of service levels and their corresponding validity constraints can be changed dynamically to reflect current network conditions at the site. Note that because EIP forces senders to use their true IP addresses, they cannot pollute these Bloom filters with spurious entries.

V. Security

Repeated DNS Queries. A sender whose traffic is restricted by a destination host may attempt to subvert the restriction by reacquiring fresh t-addressed with repeated DNS queries. EIP counters such attempts through cooperation between Internet servers (or other externally reachable hosts) and their authoritative DNS servers: repeated DNS queries from an undesirable sender will be denied or throttled.

DDoS Attacks Against DNS. Since DNS servers become "gatekeepers" for Internet servers, an attacker may attempt a (distributed) DoS attack by bombarding the authoritative DNS server with DNS queries; even if the DNS server refuses these queries to protect the host, the DNS server itself – and hence the hosts it serves – would be (D)DoSed. However, as discussed in Section IV-D, DNS servers are also reachable only through their own t-addresses, leaving only well-replicated and anycasted root DNS servers exposed. Further, access networks increasingly restrict cross-border DNS traffic to their designated resolvers for security reasons, making large-scale attempts of this kind less likely in the future.

ADNS Server Break-in. Any security component – be it a firewall or intrusion detection system or a spam filter – only works insofar as its integrity is intact. Authoritative DNS in EIP is no exception. A compromised authoritative DNS server will leave all hosts in its zone unprotected; a compromised higher-level DNS would affect all the zones beneath it. While EIP does not supplant the need to properly harden DNS servers, it facilitates their protection by controlling the traffic that reaches them. In particular, the EIP extension discussed in the previous paragraph would only allow ISP-authorized traffic to ever reach DNS servers.

Malicious LDNS. A malicious resolver may attempt to bypass EIP traffic controls by issuing spurious delegation tokens to clients, with byte sub-constraints exceeding in total the overall constraint in the destination t-address. However, since the routers police traffic based on the overall byte constraint, as the traffic from these clients converges towards the target, the aggregate validity constraint from the traget t-address will be imposed. In the meantime, if the destination receives initial type-1 datagrams with delegation sub-constraints totaling more than the overall constraint, the LDNS's misbehavior will be detected and the LDNS blacklisted. Any other deviations from proper delegation tokens (e.g., failure to increment the client counter or signing an incorrect client group) will only affect LDNS's own clients. A malfunctioning LDNS can obviously always disrupt its clients' operation.

Identity Faking and Reuse. An attacker cannot fake its identity: such a packet will be discarded at the first router. Moreover, the attacker cannot "amplify" its identity by sharing it with multiple hosts: as the traffic from these hosts converges towards the target, the aggregate validity constraint from the traget t-address will be imposed. If the attacker gains network eavesdropping capability, it can attempt a replay attack, by resending type-2 packets observed on the network, which do not carry a timestamp, or simply use a snooped FID to send its own payload. Similar to existing capability systems with unprotected FIDs [5], we consciously accept this weakness for the sake of efficiency and simply narrow the scope of these attacks through enforced validity constraints. However, with hardware-implemented signature generators becoming increasingly mainstream, in the future type-2 datagrams may include signed timestamps (and actually include the rest of the packet into the signed content for good measure), which would limit any replay opportunity to within the clock skew window.

Collusion Attacks. The attacker may obtain large capabilities from a colluding host inside the destination site and then try to saturate either the site's resources (e.g., the Internet link) or an upstream core link. While the latter can be protected in a standard way by using per-destination fair queuing at routers [5], EIP by itself does not prevent the former. Still, by forcing both ends of this communication to use their true identities, EIP makes it simpler for the site to detect and isolate its compromised hosts.

VI. Incremental Deployment

Deploying EIP requires a centralized decision by root DNS servers to add the support required by EIP as described above. After that, EIP can be introduced incrementally within the current Internet as individual providers decide to add EIP

support. An obvious approach is to use a technique similar to existing technologies for migration from IPv4 to IPv6, such as Microsoft's Teredo, which combine overlay and tunneling techniques to enable coexistence of the two Internet version. However, an elegant alternative is possible, avoiding the need for the overlay network altogether and allowing mixing and matching current IP and EIP routers at will.

Assume for a moment we can squeeze EIP datagrams into current MTU. Then, EIP can be implemented as a shim between IP and the transport layer. In the "protocol number" field of the IPv4 datagram, or the "next header" field of the IPv6 datagram, the sender will record a new number assigned to EIP. Current routers forward these datagrams normally (we verified that a new protocol header value does not affect forwarding). However, if a datagram encounters an EIP-compliant router, it will process it according to the EIP protocol. If no such router is encountered on the way to the destination, then the EIP datagram will be processed at the entrance to the destination network, or in the worst case by the destination itself.

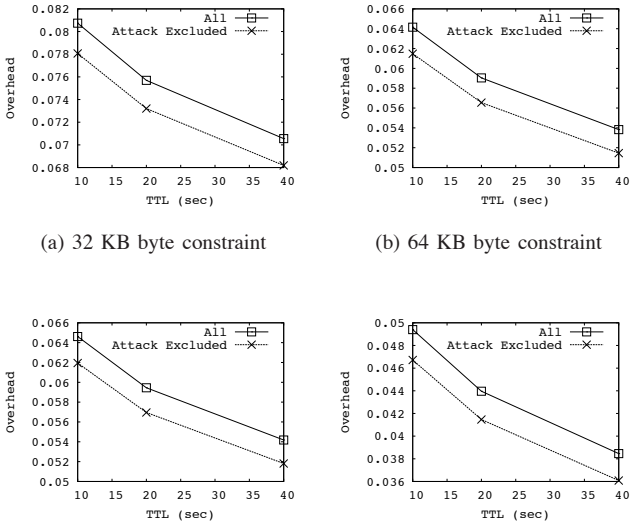
Discovering whether or not the destination supports EIP would be done in the same way as currently with IPv6: the sender's local DNS server can request both a regular A record from the authoritative server and new record type for a t-address. The EIP-compliant authoritative server may only respond with a t-address, while legacy servers would respond with a legacy A-type record.

The transition sketched above allows coexistence of EIP and legacy IP infrastructure, so that EIP end-hosts would obtain end-to-end connectivity when crossing legacy sections of the network paths. Further, every EIP-compliant component, including destinations, their edge routers, and transit routers, relegate legacy traffic to low-priority handling so that it never infringes on protected EIP traffic. The preferential treatment of EIP traffic also provides incentives for end-hosts and access networks to transition to EIP. Core networks may decide to adopt EIP since it allows them to drop malicious or unwanted traffic at ingress points - although adoption at access networks is already sufficient for EIP to achieve its effect.

What remains is to reduce the size of type-1 EIP datagram to below 1440 bytes (the 1500 MTU minus IP header – which is 40 bytes in IPv6 – and TCP headers). To achieve this, we first reduce the key length of end-hosts and DNS servers from 2048 to 1024 bits; the corresponding security strength reduction is not critical because these keys protect only a single destination network and can be frequently changed (the more valuable keys from the ISPs remain full-length). This brings the type-1 datagram header to 1442 bytes – just over our limit. As a final trick, we note that the t-addresses include source and destination IP addresses in the clear, and in the embedded datagrams they are duplicated in the IP header. Thus, we can drop them from the t-addresses saving further 32 bytes and bringing our header within the limit with a few bytes to spare.

VII. Evaluation

In this section, we first describe trace-driven simulations to assess the traffic overhead due to EIP headers and the router state size. We then compare EIP with existing capability-based approaches, in particular using a synthetic Internet-scale simulation to compare EIP with TVA. Finally, we describe our experience with the EIP prototype implementation.



(a) 32 KB byte constraint

(b) 64 KB byte constraint

(c) Byte constraint increased from 32 KB to 64 KB at first renewal

(d) Byte constraint is doubled at every renewal

Fig. 2. EIP bandwidth overhead with various time and byte constraint policies

A. Traffic Overhead

We estimate the traffic overhead imposed by EIP. We use a packet-level trace captured at the edge of a mid-sized (around 16K students, faculty, and staff) university campus on April 27, 2012. The trace represents about 5 hours of traffic exchanged between the campus network and the Internet, resulting in about 500G of captured data, which includes IP and transport-layer headers and 100 bytes of payload of each packet. We used a Dell PowerEdge 2950 server with 15K RPM hard drives and a 4-port 1Gbps Endace DAG7.5 network card for trace collection, with two ports mirroring the two utilized links on the edge router. The card counters showed no packet loss for packets reaching our instrumentation. However, router netflow statistics did show one traffic spike to 960Mbps on one of the links, and because these statistics represented 5-minute averages, it is likely that traffic exceeded our port-spanning capacity for brief intervals during this episode. We do not believe this materially affected our results.

We evaluate the traffic overhead assuming the EIP deployment as described in Section VI: each new flow in the trace incurs 1410 bytes overhead (type-1 header); subsequent packets in this flow incur 17–19 bytes (type-2 header) until the capability approaches its validity, which requires the receiver to add a continuation t-address (558 bytes with Section VI optimizations) and the sender to include this t-address in the next packet, resulting in extra 1116 bytes overhead for this connection. Our trace includes seemingly attack traffic containing isolated TCP packets to a given IP address (e.g.,

SYN floods). We calculate the EIP overhead with and without this traffic because if these packets involve fake source IP addresses, they will dropped by the first EIP-complaint router.

Figure 2 shows the overhead simulated over all traffic and over only what we consider a legitimate traffic for various capability time and byte constraints. As can be seen, increasing the byte constraint from 32 KB to 64 KB reduces the overhead significantly, from 7-8% to 5-6%, while the time constraint seems to affect the overhead somewhat less significantly. The sensitivity of overhead to byte constraint suggests more elaborate constraint provisioning. For example, a destination may choose to increase the byte constraint for a host as it builds trust about that host. This might significantly reduce the bandwidth overhead imposed by EIP. For example, Figure 2(c) shows the overhead when capability byte constraint is started with 32 KB and is increased to 64 KB upon the first renewal of the capability. Its overhead is virtually the same as when using the straight 64 KB constraint but with a better recipient control over incoming traffic. As another example, Figure 2(d) shows the overhead when the capability byte constraint is doubled every time the capability is renewed, resulting in further overhead reduction. A more careful study of constraint provisioning policies is one of the directions for future work.

B. Router State Estimation

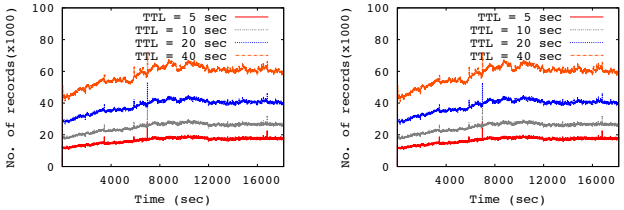
We now estimate the state that EIP routers need to maintain. We use the same trace discussed in SectionVII-A for this purpose. Thus, our results will be indicative for edge routers, which represent the "front lines" in Internet security.

We simulate the router behavior when it is presented with the packets from our trace and evaluate the number of flow records stored at the router assuming a naive approach where each new flow creates a record, which is discarded when the flow's capability constraint is reached. Figure 3 shows how the number of flow records at the router changes in the course of the 5-hour trace. Figure 3 shows that the state requirements at the edge router of a sizable intranet are manageable, e.g., they generally stay within 50K records at 20 second TTL. Given the record size of 61 bytes (16B for flow ID, 32B for the IP address pair, 9 bytes for the timestamp and 4B for the validity constraints) our edge router needs about 3MB of state – a quite manageable amount for today's line cards. From this finding, it appears that the clever trick from [5] to keep the router state below a constant, while fully applicable to EIP, might not be needed in practice, at least in the case of edge routers.

In addition, the router state overhead is affected strongly by the time constraint and is virtually independent of the byte constraint. This is understandable as a lower byte constraint for a continued flow simply replaces one flow record with another while a longer time constraint makes records linger after the communication stops.

C. Comparative Internet Scale Simulation

We compare EIP to other approaches to controlling unwanted traffic on the Internet. Previously, TVA [5] was found to offer more effective protection than SIFF [3], in which



(a) 32K Byte constraint (b) 64K Byte constraint

Fig. 3. Estimated router state for various time and byte constraints. These graphs are best viewed in color. In greyscale, the lines corresponding to smaller TTLs are closer to the X-axis indicating smaller state.

each router on the path contributes two bits to the capability construction, and Pushback [38], where routers reactively rate-limit excessive flows and propagate these limits upstream. Portcullis [6] was proposed as an answer to the denial of capability attack against an earlier version of TVA, and was shown to be advantageous compared to Speak-up [7] and TVA. EIP is not directly comparable to Portcullis or Speak-up because these two schemes require the sender to perform more work (puzzle-solving in the Portcullis case and bandwidth consumption in the case of Speak-up) than the attackers to get through to the receiver. This can cause significant delays in communication and excessive energy drainage from battery-based devices. Meanwhile, TVA answered Portcullis with an enhanced protection against denial of capability attacks through hierarchical fair queuing (HFQ) of capability requests at the routers. Consequently, the rest of this section focuses on comparing EIP with the new version of TVA as presented in [5]. The results below show that EIP offers better overall protection against large-scale attacks and requires lower level of incremental deployment to be effective.

1) Methodology: To evaluate the effectiveness of EIP in combating DoS attacks when compared to TVA, we build a discrete time event-based simulator implementing both EIP and the enhanced version of TVA [5]. We follow the general methodology from the Portcullis study [6], with most model parameters adopted from the TVA study [5]. We feed our simulator with a skitter topology [39], which represents a realistic router-level Internet scale topology as observed from a particular vantage point. It has about 115K routers. We consider the vantage point as a victim and assume for simplicity that the shortest paths spanning tree determines the routes from all the routers to the victim. While actual routes often deviate from the shortest paths, we believe this simplification does not affect our findings because at the root of these findings is a tree structure of routers rooted at the victim. To associate routers to their autonomous system number, we use the Team Cymru IP to ASN Lookup service [40]. About 3% of the routers could not be mapped to a particular ASN. We assign them the ASN of their parent in the shortest path tree.

The the victim connects to the first router with a 120Mbps link which, as in the TVA study, equals 10% of the aggregate attack bandwidth when the attack density is 80% (i.e., 80% of

the senders are attackers). Following the Portcullis study, we assume that routers in the topology are connected with 2Gbps links and traffic senders are connected to the edge routers with 20Mbps links. The bandwidth of the capability request channel is set to 5% of the link bandwidth for TVA whereas full bandwidth is available for EIP's t-address requests.

We randomly distribute a total of 1500 good and attack senders over the set of edge routers in our topology. As in TVA, all of the experiments in this section involve attackers sending capability requests at a rate of 1 Mbps. A good sender sends a fresh request for a t-address every second. Meanwhile, dropped requests are retransmitted with an exponential back-off starting from 200ms. We use a request size of 1000 bits for TVA and 1500 bytes for EIP. Like in TVA, we use an application timeout of 10 seconds, after which we assume that the application will give up resulting in communication failure. The simulation runs for 60 seconds.

We adopt the "imprecise authorization policy" from TVA in our experiment: the destination cannot reliably detect an attacker at the time of the first contact and thus grants the first capability. However, subsequent capability requests from the attacking IP address will be denied. Denying the additional t-address requests would totally stop the attackers in the EIP case since they cannot spoof their IP addresses and they cannot reach the victim's ADNS with a proper t-address from upper-level DNS servers. In the TVA case, denying the capability requests cannot prevent the attacker from reaching the victim with additional requests. By the time the victim drops these additional requests, they have already exerted the damage.

Our simulation setup differs from that of TVA in three aspects. First, we use the entire skitter topology rather than random sub-topology slices for simulation. Second, an AS in our setup can carry both access and transit traffic while in the TVA setup traffic senders can be attached only to terminal ASes. Our setup reflects the fact that most transit ISPs also offer access services. Third, an AS can host both legitimate and attack users in our setup while in TVA's setup, a terminal AS can represent either a legitimate sender or an attacker. The latter restriction is problematic because attackers and legitimate users originating from the same AS would have the same path IDs in TVA, and therefore, will share the same queues, making the legitimate users vulnerable to starvation.

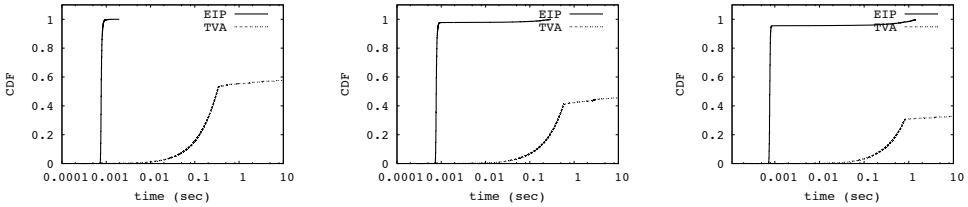
2) Results: As in the Portcullis study, we measure the degree of protection from attack traffic by the time for a legitimate user to acquire a capability request in TVA and t-address in EIP. In EIP, t-addresses are issued by victim's authoritative DNS (ADNS) server, which we assume is co-located with the victim destination, while in TVA capabilities are issued by the victim host itself. In TVA the capability requests are limited to 5% of each hop's bandwidth while in EIP the ADNS is itself protected by t-addresses, so the same attacker can only send repeated requests until they exceed the byte constraint of the ADNS's t-address: according to the imprecise authorization policy, the upper-level DNS will not issue another t-address for the attacker to contact the target ADNS. Note that in EIP, the attacker cannot spoof its IP address to bypass this protection.

In accordance with the enhanced version of TVA, every compliant router in our simulator executes hierarchical fair queuing (HFQ) to protect the capability channel: every TVA router marks each request packet with a tag and every TVA router performs HFQ on the last few tags in the request. To ensure that our model is favorable to TVA, our TVA simulator allows up to 5 path tags in a packet (an increase from up to 4 in the TVA study given that studies have shown that over 95% of the Internet paths are within five AS hops but a sizable minority of paths exceeds 4 hops [41]), unlimited number of queues, and ten slots in each queue (the largest number among the four queue levels in TVA). Furthermore, we assume that no attackers spoof their IP addresses or path tags (the latter being an additional attack avenue in TVA but not in EIP). We first run an experiment similar to the TVA's setup to verify the correct operation of our simulator. We obtained results similar to the results they obtained (these are omitted due to space limitation). Next, we start experimenting with our setup, which we believe is more realistic.

The first experiment examines the resiliency of EIP and TVA to an increasingly massive attack assuming full deployment – every router has adopted the EIP (resp. TVA) mechanism. We set the percentage of attackers to 20%, 40%, and 80%. Figure 4 shows the results when the capability byte constraint for communicating with the ADNS is set to 32 KB.

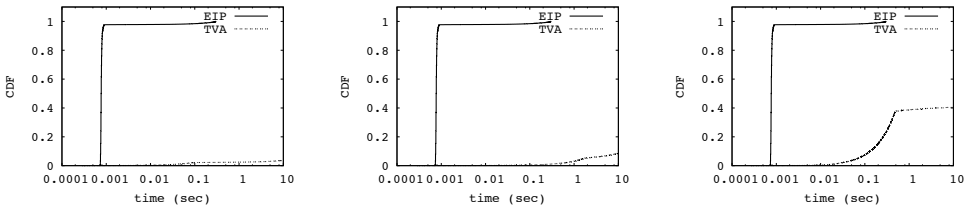
As can be seen, TVA struggles to grant legitimate users capabilities as the attacker density increases. With only 20% attackers, less than 60% of legitimate capability requests were granted within 10 second period. This percentage drops to about 35% when the attackers's density is 80%. On the other hand, EIP persistently succeeded in granting all legitimate capability requests. One reason for TVA performing worse in our experiments than in the TVA study is that we consider the full Internet topology rather than a random sub-topology. The Portcullis study pointed out previously that the full topology reduced the benefits of the original TVA; our study indicates this also holds for the enhanced TVA. Since a sub-topology in the TVA study reduces the width of the router tree, it also reduces the diversity of path IDs and hence enhances the effectiveness of HFQ. Another reason is that by allowing traffic to enter only at the leaf ASes, the attackers in the TVA study tend to have longer path IDs. When attackers can enter at an internal AS (as in our setup), they tend to have fewer tags, diminishing the effectiveness of the HFQ enhancement. Finally, unlike the TVA study, our setup models a very real possibility that both attackers and good senders may originate from the same AS. As mentioned earlier, TVA cannot distinguish these senders, making the good capability requests vulnerable to being out-crowded by the attackers. In fact, further investigation of the TVA results of our experiment shows a noticeable trend of a good user either persistently succeeding or persistently failing in obtaining capabilities throughout the simulation time: for example, in the 80% attackers case, while over 30% of requests are successful overall, 67% of the good users consistently failed over 90% of the time. This suggests that the consistently failing users might be sharing queues with attackers.

The second experiment examines the performance of TVA and EIP under partial deployment scenarios. This experiment uses similar setup as in the previous experiment except that various percentages of the ASes in the system are assumed to



(a) 20% attackers (b) 40% attackers (c) 80% attackers

Fig. 4. Capability request flood with byte constraint of 32 KByte.



(a) 20% deployment (b) 50% deployment (c) 80% deployment

Fig. 5. Capability request flood with byte constraint of 32 KB under partial deployment with 40% attackers.

have not deployed EIP or TVA. In particular, we assume that the victim's AS has deployed the scheme (TVA or EIP). In addition, 20%, 50%, or 80% of the remaining ASes chosen randomly deploy the scheme. This experiment features 40% of senders being attackers and an ADNS capability byte constraint of 32 KB. Figure 5 shows the results, which indicate that TVA requires wide-spread deployment to achieve tangible protection. This behavior is expected since non-complaint routers do not contribute to the path ID as required by TVA, which results into a larger group users (legitimate and attackers) sharing queues. With EIP, legitimate users achieve 100% request satisfaction even with modest deployment.

D. Prototype Implementation

We prototyped[7] the EIP host and router using the Linux netfilter framework [42] to intercept and process packets, with packet processing done in user space, resulting in a network layer that could support unmodified TCP applications at the end hosts. The main goal was to verify our design and have a preliminary assessment of the performance at the end-hosts (since our user-space software implementation of the router is not indicative of a realistic router). This in particular yielded useful insights for end-host system design that interacts better with the TCP layer above. Specifically, to avoid interaction with TCP's RTO estimation, a host maintains an asynchronous thread that ensures there is a continuation t-address for ready delivery when needed; further, the continuation t-address is sent once the flow reaches a certain threshold in terms of incoming bytes, again in advance of the immediate need. In real deployment, selection of threshold values would be another policy decision. In our experiments, we select threshold values that avoid triggering TCP retransmission. The prototype involved 1542 new lines of C code for the end-hosts and 1090 for the routers.

Our testbed is a host-router-host configuration of three commodity Dell R610 machines[8] running 3.2.0-23-generic kernel with Broadcom NetXtreme-2 gigabit NICs. Cryptographic operations are performed using the OpenSSL 1.0.1 library [43]. All signatures are 1024-bit RSA, which represents a compromise between key length, data size, and algorithm speed. The t-addresses in these experiments have validity constraints of 10 sec. and 1MB.

We evaluate throughput using unmodified legacy TCP applications, curl and Apache, running on the end-hosts. Given a 9000 byte jumbo MTU in our setup, our host implementation decreases it to 6500 bytes, thereby allowing a maximum of 2500 bytes for the addition of EIP headers. For comparison, we use a "vanilla" implementation that processes identical file transfers. Our vanilla implementation utilizes the same MSS modification to allow for fair comparison. Figure 6 shows throughput as reported by curl when downloading files of different size. EIP overhead is higher for small transfers such as a DNS query or ad download but is amortized over large or long-running transfers such as streaming media or pipelined HTTP. In Figure 6 we see EIP throughput is just over 32% of vanilla throughput for transfers of 10KB but gradually

[7]Our datagram description in Section IV-C reflects minor fine-tuning after the implementation of the prototype, including a switch from the absolute to relative expiration times and elimination of flow TTL. These changes are inconsequential to our prototype evaluation.

[8]Two of the machines – the router and one of the hosts – were 4-core Xeon 2.0GHz CPU with 6GB 1333 MHz memory and the third machine was 2x8 Xeon 2.40GHz CPU with 8GB 1066 MHz memory.

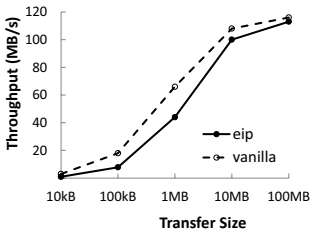


Fig. 6. EIP Throughput

TABLE I. EIP ROUTER PROCESSING OVERHEAD

	min	mean	median
type-1	161μs	198μs	200μs
type-2	11ns	74ns	61ns

increases to 97% of vanilla throughput for transfers of 100MB. The low RTT (<1ms) in our testbed exaggerates the impact of the cryptographic processing delay on throughput – in real deployment, this delay would be masked by higher RTTs as well as by hardware-implemented verification at the routers.

We evaluate processing overhead at the router for both types of EIP datagram. We run file transfers through our router to generate thousands of each datagram type. We count the instruction cycles used to process each datagram in accordance with Intel recommendations for benchmarking C in Linux [44]. Our user-space code is subject to interrupts, so we present the minimum (indicative of inherent processing costs), mean, and median processing time for each datagram.

Table I shows the results. Type-2 processing incurs only 11ns of inherent processing, which is important because these datagrams carry most of data transfers and also because it bears more similarity with a real implementation since it does not involve crypto operations. Type-1 processing, at 160μs is impractically high, but we verified that it is virtually entirely due to OpenSSL calls for four RSA verifications per packet. A hardware implementation in a real router should greatly reduce this overhead.

VIII. FUTURE WORK: POLICIES

An important direction for future work is policies and mechanisms for setting dynamic validity constraints for t-addresses. This includes the design of a flexible rule language to express the policies, the protocol between destinations and their authoritative DNS servers to communicate the policies, and the techniques for derivation of the policies themselves.

A sender can be assigned a t-address with varied validity constraints (or be rejected altogether) depending on its past behavior. If past extends beyond the immediate interaction, this involves maintaining client ratings, which is somewhat similar to reputation in reputation systems except its scope is limited to the destination site. Because of that, it is free from typical issues in reputation systems, such as data pollution, privacy, and trust. In principle, we could envision client ratings to be shared among authoritative DNS servers, or further disseminated from authoritative name servers up the registration path, all the way to the root servers. In this case, client ratings would no longer be local to the destination site as the higher level name servers would coalesce these ratings from multiple destination sites. Client ratings would then converge to a traditional reputation system and would have to address the typical issues above.

IX. RELATED WORK

Our transient addresses are in essence capabilities. Existing capability systems [5], [6], [3], [45] include an unprotected communication channel to hosts for initial capability requests; these requests reach hosts using their IP addresses. In EIP, DNS (itself protected by EIP except for the roots that are defended by anycast) is used to distribute capabilities, thus end-hosts are never exposed. Consequently, EIP provides better protection at the cost of requiring certain support from ICANN and root DNS servers. The SANE architecture [46] uses separate network components to issue capabilities in the context of enterprise networks. By operating at a higher layer, EIP offers less hard-and-fast protection but targets public Internet and does not require separate components. In fact, our approach does not preclude a SANE deployment within an enterprise. Several proposals design capability systems for wireless networks [47], [48]. In particular, the DIPLOMA system [47] never sends capability requests to the destinations but—similar to SANE—requires the sender to obtain the initial authorization token from a new infrastructure component; we leverage DNS for this purpose. A survey and taxonomy of capability systems can be found in [49].

Filtering approaches ([50], [51]) 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 [50]. Pushback (e.g., [38], [52], [53]) is another filtering approach where routers propagate rate-limiting rules from recipients towards sources of heavy traffic. OffByDefault [54] extends this notion further by propagating arbitrary filtering rules proactively. EIP does not require a new signaling or rule propagation protocol or filtering rules maintenance by the routers.

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 out the vulnerability of capability systems to the *denial of capability* attack, where the attackers exhaust the unprotected capability request channel [4]. Parno et al. countered with an elegant scheme using proof-of-work to ensure that legitimate capability requests will eventually get through [6]. 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 [50]. By relegating capability assignment to DNS, our approach narrows the scope of vulnerability to the root DNS servers, which are protected through Internet-wide anycast deployment.

A number of overlay approaches leverage the capacity of overlay nodes to protect hosts from DDoS attacks (e.g., [55], [28], [29], [56]). In contrast, EIP attempts to build generic security protections at the IP layer.

Several approaches improve security through separate rendezvous points [57], [58], a name-routed signaling infrastructure [59], or by isolating client and server address spaces

[60]. EIP involves less significant architectural changes; instead of introducing an extra indirection point, it leverages already existing indirection stemming from name resolution. The Host Identity Protocol [11] separates host identifiers from IP addresses and distributes the former through DNS. EIP goes beyond this as it distributes capabilities as *transient* addresses and enables in-network host authentication. Shue et al. [61] proposed to rotate through a set of IP addresses for a host, of which only a small subset of addresses are valid at any given time, and selectively resolving DNS queries to currently valid IP addresses thus controlling access to the host. Phatak et al. [62] extend this idea to the client side, thereby providing a degree of anonymity to the clients. EIP shares with these approaches their reliance on DNS but allows for finer-grained, volume-limited, and cryptographically enforced traffic control.

Several approaches address network security by enabling packet source attribution and accountability [63], [64], [65]. In particular, IPa+ [65] offers an alternative to RPKI for IP address authentication. At the same time, Clark and Landau [23] question the utility of stronger attribution than what is already possible with IP addresses in the face of multi-step attacks. Our approach does not introduce a new privacy paradigm but inherits the paradigm of the overall network architecture. In particular, as described here, EIP preserves the privacy characteristics of the current Internet. Seehra et al. provide a general argument for the desirability of allowing recipient's control over incoming communication [66]. EIP studies a design that would achieve this goal yet be able to coexist with the current Internet.

X. Conclusion

Security and privacy are perennial challenges in network design, and are often at contradiction with each other. This paper offers a new approach to addressing these challenges. It affords destination networks full control over the traffic that is to reach them and makes it possible to drop other traffic early on its journey ("pushing the firewall into the network" [67]). While the desirability of these functions has been well established, our approach achieves them without introducing any new signaling protocols, without requiring routers to maintain and propagate through the network complex filtering rules, and without affecting the privacy properties of the network. In particular, as described here, it preserves the privacy properties of today's Internet. Thus, our approach explores the minimal intervention into today's Internet architecture that would allow it to support the above functions, and in fact of its mechanisms are deployable within today's Internet. We further hope that the results of this research will impact the parallel efforts on future Internet architectures [1].

The code developed for this project is available at [68]. We thank Xiaowei Yang of Duke University for great tips on fast flow cache lookups and Linux cycle counting.

References

[1] "NSF Future Internet Architecture project," http://www.nets-fia.net/.

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

[3] 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.

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

[5] X. Yang, D. Wetherall, and T. Anderson, "TVA: a DoS-limiting network architecture," *IEEE/ACM ToN*, vol. 16, no. 6, pp. 1267–1280, 2008.

[6] 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.

[7] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, "DDoS defense by offense," in *SIGCOMM*, 2006.

[8] "European Future Internet Portal," www.future-internet.eu.

[9] M. Rabinovich and O. Spatscheck, "Evasive internet: Reducing internet vulnerability through transient addressing," in *IEEE Global Internet Symp.*, 2010.

[10] D. Meyer, "The locator identifier separation protocol (LISP)," *The Internet Protocol Journal*, vol. 11, no. 1, March 2008.

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

[12] E. Nordmark and M. Bagnulo, "Shim6: Level 3 multihoming shim protocol for IPv6," Request for Comments 5533.

[13] M. Lepinski and S. Kent, "An infrastructure to support secure internet routing," Request for Comments 6480, 2012.

[14] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing," RFC 2827, May 2000.

[15] F. Baker and P. Savola, "Ingress filtering for multihomed networks," RFC 3704, March 2004.

[16] R. Beverly, A. Berger, Y. Hyun, and k. claffy, "Understanding the efficacy of deployed Internet source address validation filtering," in *SIGCOMM IMC*, 2009.

[17] S. M. Bellovin, "Security problems in the TCP/IP protocol suite," *SIGCOMM Comput. Commun. Rev.*, vol. 19, pp. 32–48, April 1989.

[18] U. C. E. R. Team, "Vulnerability note VU#800113. multiple DNS implementations vulnerable to cache poisoning," http://www.kb.cert.org/vuls/id/800113.

[19] V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks," *CCR*, vol. 31, no. 3, pp. 38–47, 2001.

[20] J. Touch, "Defending TCP against spoofing attacks." RFC 4953, 2007.

[21] "The Honeynet Project. Know your enemy: Fast-Flux Service Networks," http://www.honeynet.org/papers/ff/, July 2007.

[22] T. Holz, C. Gorecki, K. Rieck, and F. Freiling, "Measuring and detecting fast-flux service networks," in *NDSS*, 2008.

[23] D. Clark and S. Landau, "The problem isn't attribution; it's multi-stage attacks," in *ReArch*, 2010.

[24] M. Konte, N. Feamster, and J. Jung, "Dynamics of online scam hosting infrastructure," in *PAM*, 2009, pp. 219–228.

[25] K. Poulsen, "FBI busts alleged DDoS mafia. SecurityFocus." http://www.securityfocus.com/news/9411.

[26] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: characterization and implications for CDNs and Web sites." in *WWW*, 2002.

[27] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds," in *NSDI*, 2005.

[28] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: secure overlay services," in *SIGCOMM*, 2002.

[29] D. G. Andersen, "Mayday: distributed filtering for Internet services," in *USITS*, 2003.

[30] M. Patrick, "DHCP relay agent information option," RFC 3046, 2001.

[31] R. Droms and W. Arbaugh, "Authentication for DHCP messages," RFC 3118, June 2001.

[32] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching," *Networking, IEEE/ACM Transactions on*, vol. 10, no. 5, pp. 589–603, 2002.

[33] H. Qian, M. Rabinovich, and Z. Al-Qudah, "Bringing local DNS servers close to their clients," in *IEEE GLOBECOM*, 2011, pp. 1–6.

[34] R. D. Graham, "No, #Anonymous can't DDoS the root DNS servers," http://erratasec.blogspot.com/2012/02/no-anonymous-cant-ddos-root-dns-servers.html.

[35] V. Paxson, "End-to-end routing behavior in the internet," *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 601–615, 1997.

[36] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," in *SIGCOMM IMW Workshop*, 2002.

[37] "NITROX XL CNN35XX Security Adapter Family," http://www.cavium.com/pdfFiles/Nitrox-XL_CNN35XX_Rev1.0.pdf?x=1.

[38] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, pp. 62–73, 2002.

[39] CAIDA, "Skitter," http://www.caida.org/tools/measurments/skitter/.

[40] "Team Cymru Community Services IP to ASN Mapping." http://www.team-cymru.org/Services/ip-to-asn.html.

[41] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, k. c. claffy, and A. Vahdat, "The Internet AS-level topology: three data sources and one definitive metric," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 17–26, Jan. 2006.

[42] "The netfilter.org project," http://netfilter.org.

[43] "Welcome to the OpenSSL Project," http://www.openssl.org.

[44] G. Paoloni, "How to benchmark code execution times on intel ia-32 and ia-64 instruction set architectures. Intel White Paper," http://download.intel.com/embedded/software/IA/324264.pdf, 2010.

[45] X. Liu, X. Yang, and Y. Xia, "Netfence: preventing internet denial of service from inside out," in *ACM SIGCOMM Conference*, 2010.

[46] 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.

[47] M. Alicherry and A. D. Keromytis, "Diploma: Distributed policy enforcement architecture for MANETs," in *4th IEEE Int. Conf. on Network and System Security (NSS)*, 2010, pp. 89–98.

[48] E. Swankoski and S. Setia, "EPIC: Efficient path-independent capabilities in mobile ad hoc networks," in *IEEE Int. Workshop on Security and Privacy of Mobile, Wireless, and Sensor Networks (MWSN)*, 2013.

[49] V. Kambhampati, C. Papadopoulos, and D. Massey, "A taxonomy of capabilities based DDoS defense architectures," in *9th IEEE/ACS Int. Conference on Computer Systems and Applications (AICCSA)*, 2011.

[50] X. Liu, X. Yang, and Y. Lu, "To filter or to authorize: Network-layer DoS defense against multimillion-node botnets," in *SIGCOMM*, 2008.

[51] K. Argyraki and D. Cheriton, "Scalable network-layer defense against internet bandwidth-flooding attacks," *IEEE/ACM Trans. Netw.*, vol. 17, pp. 1284–1297, August 2009.

[52] T. Peng, C. Leckie, and K. Ramamohanarao, "Defending against distributed denial of service attack using selective pushback," in *IEEE Int. Conf.on Telecomm.*, 2002.

[53] V. Foroushani and A. Zincir-Heywood, "TDFA: Traceback-based defense against DDoS flooding attacks," in *IEEE 28th Int. Conf. on Advanced Information Networking and Applications (AINA)*, May 2014.

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

[55] C. Dixon, T. Anderson, and A. Krishnamurthy, "Phalanx: Withstanding multimillion-node botnets," in *USENIX/ACM NSDI*, 2008.

[56] H. Wang, Q. Jia, D. Fleck, W. Powell, F. Li, and A. Stavrou, "A moving target DDoS defense mechanism," *Comp. Comm.*, vol. 46, 2014.

[57] D. Adkins, K. Lakshminarayanan, A. Perrig, and I. Stoica, "Towards a more functional and secure network infrastructure," http://i3.cs.berkeley.edu/publications/papers/csd-03-1242.pdf, UC Berkeley, Tech. Rep. UCB/CSD-03-1242, 2003.

[58] A. Gurtov, D. Korzun, A. Lukyanenko, and P. Nikander, "Hi3: An efficient and secure networking architecture for mobile hosts," *Comp. Comm.*, vol. 31, no. 10, pp. 2457 – 2467, 2008.

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

[60] M. Handley and A. Greenhalgh, "Steps towards a DoS-resistant Internet architecture," in *FDNA (ACM SIGCOMM Workshop)*, 2004.

[61] C. A. Shue, A. J. Kalafut, M. Allman, and C. R. Taylor, "On building inexpensive network capabilities," *CCR*, vol. 42, no. 2, pp. 72–79, 2012.

[62] D. Phatak, A. T. Sherman, N. Joshi, B. Sonawane, V. G. Relan, and A. Dawalbhakta, "Spread Identity: A new dynamic address remapping mechanism for anonymity and ddos defense," *Journal of Computer Security*, vol. 21, no. 2, pp. 233–281, 2013.

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

[64] 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.

[65] X. Yang and X. Liu, "Internet protocol made accountable," *ACM HotNets-VIII*, 2009.

[66] 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.

[67] T. Anderson, K. Birman, R. Broberg, M. Caesar, D. Comer, C. Cotton, M. Freedman, A. Haeberlen, Z. Ives, A. Krishnamurthy, W. Lehr, B. Loo, D. Mazieres, A. Nicolosi, J. Smith, I.Stoica, R. van Renesse, M. Walfish, H. Weatherspoon, and C.Yoo, "NEBULA - a future Internet that supports trustworthy cloud computing," http://nebula.cis.upenn.edu/NEBULA-WP.

[68] "EIP code," http://engr.case.edu/rabinovich_michael/eip.html.

Zakaria Al-Qudah Zakaria Al-Qudah received his B.Sc. degree from Yarmouk University, Jordan, and his M.Sc. and Ph.D. degrees in computer engineering from Case Western Reserve University. He is currently an associate professor of computer engineering at Yarmouk University. His research interests are in the area of Internet systems, measurements, and security.

Eamon Johnson Eamon Johnson received the B.S. degree in computer science from the University of Illinois at Urbana-Champaign in 2000 and the M.S. degree in computer science from Case Western Reserve University in 2012, where he is now candidate for the PhD degree. His research interests include network security and text mining.

Michael Rabinovich Michael Rabinovich is a professor in the EECS department at CWRU, which he joined in 2005 after spending 11 years at AT&T Labs—Research. He holds a PhD from University of Washington. His research revolves around Internet architectures, performance and security.

Oliver Spatscheck Dr. Oliver Spatscheck a researcher at AT&T Labs—Research. He received his PhD in computer science from the University of Arizona. His general interests are network-centric systems, network measurements and network security.