Secure Availability Monitoring Using Stealth Probes

Ioannis Avramopoulos[*], *Dimitris Syrivelis*[†], *Jennifer Rexford*[*], *and Spyros Lalis*[†]
Princeton University[*] *and University of Thessaly*[†]

Abstract

IP routing protocols naively assume that routers are trusted, egregiously failing when routers become adversarial. Because of the significant power they give to adversaries, routers are an increasingly attractive target for subversion attacks. In this paper, we present the design, implementation, and evaluation of stealth probing, a secure path-availability monitor that prevents on-path adversaries from degrading user performance. Stealth probing selects data packets to serve as implicit probes that should be explicitly acknowledged. The probes are concealed, preventing adversaries from treating them preferentially. Secure monitoring enables failure recovery by signaling when networks should reroute traffic to alternate paths. Testbed measurements of our software prototype demonstrate that stealth probing has practical overhead comparable to destination-based forwarding.

1 Introduction

Anecdotal evidence suggests that router compromises, either from remote attackers or disgruntled network operators, have become an operational reality [23]. Secure routing protocols do not solve the problem as they do not prevent on-path attackers from manipulating the data traffic and off-path colluding attackers from faking direct connectivity to attract data traffic. Upon reaching a compromised router, data packets can be discarded by configuring packet filters on the router or tunneled to a remote location for eavesdropping, modification, and impersonation attacks. End-to-end encryption can thwart the latter two attacks but falls short on the first. A data-plane adversary that can arbitrarily drop packets is especially troublesome because these attacks are so difficult to diagnose and can be used to block communication at critical times (such as natural disasters or physical attacks).

In this paper, we devise mechanisms that prevent disruptions in end-to-end availability. We present *stealth probing*, a secure path-availability monitor that measures the performance of paths in a fashion that prevents even an on-path attacker from evading detection. Such monitoring enables failure recovery by triggering the network to reroute traffic to alternate paths. Alternate paths can be selected using an Intelligent Route Control system [1, 12] or multipath interdomain protocols such as MIRO [33] and ACR [32]. In fact, recent research (e.g., [32]) has advanced availability as the *only* property that an interdomain routing system needs to provide; secure monitoring is vital to such systems.

1.1 Threat Model

Networks can be attacked from either hosts or routers. Although routers are presumably harder to compromise, the possible damage of attacks from within the infrastructure goes far beyond the impact of remote attacks from the edge. We consider attacks from hosts but focus on control-plane and data-plane attacks launched from compromised routers. In a control-plane attack, the adversary falsifies routing information aiming, for example, to deflect traffic onto his routers. Data-plane attacks manipulate the received traffic. The goal in a data-plane attack can be to eavesdrop, impersonate destinations, or disrupt communication. Although we consider all of these attacks, we focus primarily on the latter.

Disrupting end-to-end communication can be as simple as installing an access-control list on the router to selectively discard the victim traffic, while allowing probe traffic through. The effectiveness of the attack derives from the difficulty in diagnosing it using standard monitoring tools. In addition to discarding the traffic, the adversary can disconnect the destination by delaying, modifying, and reordering the traffic, by injecting forged traffic, and by replaying old traffic. Although most routers cannot directly perform these attacks at line rate, the victim traffic can be deflected to remote hosts (such as a botnet), forming a *data wormhole*. The adversary could

use data wormholes to evade detection from naive monitoring techniques. In addition, rather than discard all data packets, the adversary may degrade performance to render the path effectively unusable. A secure monitoring system should be robust to all of these attacks. Section 2 discusses these requirements in more detail and points out the limitations of existing solutions.

1.2 Stealth Probing

Stealth probing is a secure availability monitor based on a measurement methodology that we call "passive probing." In passive probing, data packets between the ingress and egress routers of a path are passively selected (or *sampled*) by both routers to serve as implicit probes. Probes should be acknowledged by the egress router. Using the acknowledgments, the ingress router can measure losses and round-trip delays. Stealth probing is a secure passive-probing protocol that prevents an on-path adversary from determining which data packets are probes and, therefore, prevents the adversary from treating the probe traffic preferentially. Sampling ensures that performance measurements are accurate and that probing is unobtrusive. The sampling rate can be chosen to balance accuracy and overhead according to the monitoring requirements of individual networks.

In designing, implementing, and evaluating stealth probing, we make the following contributions:

Lightweight passive probing between edge routers: Stealth probing strikes a careful balance between techniques for passive and active monitoring to provide provably accurate estimates of path performance, even in the presence of on-path adversaries. Placing key functions, such as hash-based packet sampling and packet encryption, at the edge routers leads to accurate measurements of path performance in an efficient manner even if on-path adversaries control the data packets.

Deployment incentives through additional benefits: Stealth probing is designed as an extension of IPsec to offer benefits beyond secure monitoring. Secure tunnels between edge routers can also protect host-to-host communication, prevent traffic analysis attacks, and prevent spam and denial-of-service attacks. The tunnels also simplify interdomain traffic engineering and network troubleshooting.

Prototype implementation in Click modular router: Our prototype of stealth probing in Click [19] includes several optimizations for high performance. For example, the sampling process leverages the output of the keyed hash used to authenticate the packets, and the security association for each destination prefix is stored directly in the forwarding table. Consistent hashing is used to split traffic over multiple paths to prevent packet reordering in TCP flows. Performance data are collected through a combination of per-probe logging in the kernel and aggregation of traffic statistics in user space.

Performance evaluation in a small-scale testbed: The evaluation of our prototype in a four-node testbed illustrates that stealth probing can run at high speeds, even on a software router. Stealth probing adds just a few tens of microseconds of processing delay to the data packets. The overhead of the acknowledgment packets has only a slight effect on TCP throughput, due to the small amount of extra bandwidth consumed. Overall, stealth probing introduces only a small overhead beyond basic packet encapsulation and decapsulation.

These contributions are discussed in Sections 3 through 6, respectively. Then, we discuss related work in Section 7 and conclude the paper in Section 8.

2 Secure Availability Monitoring

This section articulates the requirements for secure availability monitoring and explains why existing methods cannot adequately solve this problem.

2.1 Definitions and Requirements

We start by defining *availability*. We say that a path is *available* if end hosts can make use of this path to communicate. According to this definition, the availability of a path can be determined by the performance requirements of the end hosts and the performance characteristics of the path. Common performance characteristics are the loss rate, delay, jitter, and available bandwidth. *Availability monitoring* is a procedure that determines the availability of a path by measuring the performance characteristics of the path. We say that an availability monitoring procedure is *secure* if an adversary cannot "significantly" affect the *accuracy* of the measurements.

The essential requirement for secure availability monitoring is that an adversary cannot render the measurements inaccurate. For example, if the monitoring procedure measures a loss rate of 1% whereas the actual loss rate is 100%, then security has been breached. The monitoring procedure should be able to set limits on a permissible violation of the measurement accuracy (therefore, bounding the "significance" of the impact of an attack). An ideal monitoring procedure would measure performance exactly; however, this is typically impossible in practice, even in the absence of an adversary. Network measurements typically employ sampling (using, for example, probes) and an associated statistical inference methodology. Statistical errors are unavoidable in this setting, making exact measurements impossible. Hence, the goal of secure availability monitoring is to have a tight bound on the estimation error, even if an adversary lies along the monitored path.

2.2 Limitations of Existing Solutions

In the rest of the paper, we focus on the measurement of loss and delay. Methodologies for measuring these quantities can be classified as *passive* and *active*. Passive measurements involve the direct observation of cross traffic at the routers whereas active measurements involve sending probe packets and observing the probe replies. We analyze below the limitations of existing such techniques for secure monitoring, starting with active measurements because of their widespread adoption.

Active measurements involve probes sent from one host to another using ICMP packets (e.g., echo requests and replies), UDP packets (e.g., DNS queries and responses), or TCP packets (e.g., `http` downloads). Active probing has been extensively used in Intelligent Route Control (IRC) systems [1, 12] to avoid failures and balance traffic to improve performance and reduce cost. However, an adversary on the data path can easily distinguish the probe packets from the normal traffic and treat the probes preferentially (e.g., by delivering the probe traffic while discarding the non-probe packets). The probes may be identified based on the protocol number (e.g., ICMP) or the source IP address of a dedicated probe machines, as well as other information such as the size or timing of probes. We do not expect that the probes can be made indistinguishable while still retaining their capability to measure the performance of IP paths of the data traffic (which precludes employing overlay-based anonymizing techniques) without the direct involvement of the routers.

Passive measurements inside a routing domain are routinely performed using SNMP. However, using SNMP for monitoring interdomain paths would require coordination between ISPs, which would impose a significant administrative burden. In an interdomain setting, passive measurements involve monitoring TCP and UDP flows. Monitoring is best performed at edge routers that can observe both directions of the traffic, though certain measurements can be peformed by observing only one direction. Listen [29] is an example of a protocol that uses passive observations of TCP traffic to determine *reachability*, which is the simplest form of availability monitoring. Passive observations of TCP traffic have several limitations for monitoring availability. First, they require the monitor to maintain per-flow state. Second, if TCP traffic is not protected end-to-end, then an adversary can disrupt connectivity and evade detection by impersonating the destination. Furthermore, end-to-end encryption would prevent per-flow monitoring altogether. Finally, passive monitoring of individual flows cannot easily distinguish between server and network failures.

Figure 1: End-routers of a network path, hosts served by those routers, and IP paths connecting the routers.

3 Stealth Probing

Stealth probing is a secure availability-monitoring protocol operating between *edge routers* that combines aspects of both passive and active measurement. Considering the paths from an ingress router to an egress router (see Figure 1), stealth probing relies on the following three processes:

A **sampling process** that selects a subset of the packets crossing the path to serve as implicit probes. The output of this process becomes available to both the ingress router, which expects an ACK for each probe, and the egress router, which replies with an ACK. By keeping a timestamp for each probe, the ingress router can measure both packet loss and round-trip delays.

A **concealment process** that prevents an on-path adversary from distinguishing between probe and non-probe packets. This process precludes the preferential treatment of probe packets by the adversary. ACK packets need not be concealed, since maliciously dropping or delaying an ACK would only enhance the ingress router's ability to identify the path as faulty.

An **integrity assurance process** that prevents an adversary from inducing inaccurate measurements by modifying traffic, injecting forged traffic, or replaying old traffic (either data packets or ACKs).

In this section, we explain and justify our design decisions for realizing these three processes. In our design, the edge routers perform sampling by applying a hash function to certain fields of each packet; if the image of the hash is less than a threshold, the packet is treated as a probe. Concealment is achieved by encrypting all packets, to prevent selective attacks against a small subset of the traffic (e.g., a particular pair of communicating hosts). Ingregrity assurance is achieved through authentication. We achieve both concealment and integrity assurance by using IPsec tunnels between the edge routers for backward compatibility with existing IPsec implementations; using IPsec offers additional deployment incentives for stealth probing, as discussed in Section 4.

3.1 Monitoring from Edge Routers

Our design of stealth probing places the key monitoring functions at the edge routers that connect a stub network to the rest of the Internet. An interesting alternative

would be to have end hosts perform their own monitoring, to enable greater customization of the measurements to specific applications. However, the majority of IP traffic is generated by a narrow set of applications, such as Web, e-mail, and VoIP. Therefore, we argue that monitoring does not need to be highly customizable to be effective. Furthermore, router-based monitoring has the following advantages over host-based probing:

- The edge router sees a much higher volume of traffic, enabling fast, accurate detection of performance problems; in contrast, each host sees only a small subset of the traffic, making detection slower.
- The edge router can react to the measurement results by shifting traffic away from a faulty path; in contrast, each end host would need to perform failure recovery on its own, leading to extra overhead.
- In shifting traffic to new paths, the edge router can manage the aggregate traffic demands with the organizations load-balancing policies in mind; in contrast, end hosts typically do not have sufficient visibility or control to take corrective action.
- The ingress router can identify reachability problems along the network path to the egress router; in contrast, a pair of communicating hosts cannot easily distinguish between network and host failures.
- Monitoring between edge routers requires only one security association for each pair of stub networks; in contrast, managing security associations for all end hosts would be complicated, and make the monitoring system vulnerable to compromised hosts.

As such, we focus our attention on extensions to edge routers to perform secure availability monitoring.

3.2 Hash-based Passive Probing

Stealth probing employs a hybrid of active and passive measurement to reduce overhead and compute accurate inferences of path performance. Rather than sending explicit probe packets, the edge routers sample a subset of the data traffic already traversing the path. To ensure both routers sample the same packets, the sampling is *pseduo*random, using a hash function. The ingress and egress routers apply the same hash function to a subset of bytes in the data packet, ignoring the bytes that change as the traffic traverses the network (e.g., the IP checksum and TTL fields) or have very low entropy (e.g., the IP version number), as in earlier work on trajectory sampling [9]. If the image of the hash falls below a predetermined threshold, the packet is treated as an implicit probe; otherwise, the packet is a not a probe. The hash function must be chosen in a way that prevents the adversary from learning the hash. For example, the edge routers could sample using a keyed hash function or encrypt all of the data packets to ensure the adversary cannot distinguish between probe and non-probe traffic.

Having the ingress routers send active probes would have been interesting, but less attractive, alternative, for several reasons:

- Active probes introduce extra traffic that consumes resources along the path; at high probing rates, the active probes may, in fact, interfere with the very properties they are designed to measure.
- Concealing the active probes is more difficult. The ingress router could conceivably conceal the probe traffic by encrypting all of the packets, both probes and non-probes. However, the adversary may be able to distinguish the probe packets by their size, or the inter-arrival times, forcing the ingress router to pad the packets and randomize the timing.
- Determining the appropriate probe rate is hard. To ensure the probes stay below some fraction of the total traffic, the ingress router would need to adjust the rate of active probes to the fluctuations in the volume of data traffic. In passive sampling, treating a fixed percentage of the packets as probes is trivial.

The one advantage of active probing is the ability to measure alternate paths that are not currently in use, albeit with the risk that the adversary can readily identify these probe packets. Instead, we envision that, in stealth probing, the edge routers would split the data traffic over multiple paths, enabling passive sampling of each path. Splitting the traffic over multiple paths is attractive for a variety of other reasons, including more flexible load-balancing policies and limiting the ability of a single adversary to see all of the traffic.

For each passively-sampled data packet, the egress router sends an acknowledgment packet back to the ingress router. In theory, we could reduce the overhead of stealth probing by instead having the egress router send periodic reports about a *window* of probe packets. However, periodic snapshots would lead to slower detection of packet losses and would not enable accurate estimates of round-trip times, making it difficult for the ingress router to react to attacks (such as data wormholes) that introduce delay. Although the explicit acknowledgment packets introduce overhead, these packets are much smaller than the average data packet, limiting the overhead. In addition, the egress router does *not* need to conceal the acknowledgment packets. The adversary has no incentive to drop or delay the acknowledgment packets, since these actions would only make the ingress router more likely to detect that the path has a performance or reachability problem.

3.3 Performance Inference from Samples

Sampling at low enough rates makes the network overhead of stealth probing practical. In this section, we answer the question of how low to choose the sampling rate so as to achieve a target accuracy. Rather than present a definitive sampling rate, we present a method for making informed decisions about it according to the individual requirements of networks. The answer is based on a method for infering performance.

Stealth probing performs secure measurements by choosing data packets to serve as implicit probes and preventing an adversary from distinguishing the probes; if the adversary discards or delays data packets, the implicit probes will be inevitably discarded or delayed as well. Therefore, assuming that an adversary is on the monitored path, the measurements from the probes represent an accurate *statistical sample* of the behavior of the adversary. This enables us to infer performance by using statistical sampling theory [30]. Vis-a-vis active probing that infers performance using indirect measurements and theorems on how those measurements correlate with the statistical behavior of the system (such as PASTA), passive probing is more accurate because measurements are directly performed on the population of packets. In fact, probing based on PASTA was recently shown to be suboptimal and, in several cases, inaccurate [7].

We will illustrate the statistical sampling method for inferring performance in passive probing with a specific example: Consider a network deploying stealth probing in an outbound path and suppose that the objective is to decide whether the loss rate of this path is above or below a loss-rate-threshold (raising an alarm if it is above). A packet is considered lost if an ACK is not received or if the round-trip delay exceeds a delay-threshold. One method for making the decision is to consider a window of packets, divide the number of losses in the window with the window size, and compare the result to the loss-rate-threshold. Although simple, this test provides little control over the false alarm rate and, in fact, one can show that this test produces a high rate of false alarms.

In an alternative approach, the network operator can set an upper bound on the false alarm rate and maximize the probability of correctly detecting failures. Fig. 2 shows the accuracy of such tests (called Uniformly Most Powerful tests of a Neyman-Pearson hypothesis testing problem [21]) as the sampling rate varies. The probability of a false alarm is 5%, the observation window is $10,000$ packets and the threshold is $1,000$ packets (i.e., threshold for loss rate is 10%). We observe that there are diminishing returns by increasing the sampling rate and that there is little value in increasing the sampling rate above 2%.

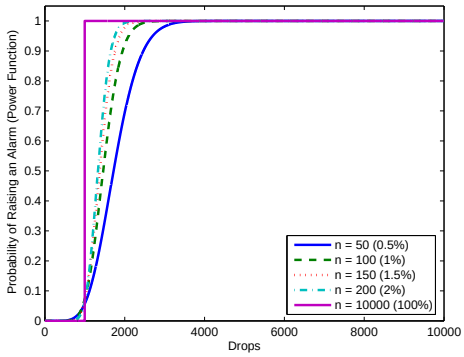


Figure 2: Detection accuracy as the sampling rate varies.

3.4 Data-Packet Encryption

The primary reason for encrypting data packets is not to conceal the probes (note that there are other techniques for implementing concealment) but to protect against attacks that target the performance inference method. Consider, for example, an unencrypted stealth probing session to prefix `10.92.0.0/16`. If the session traffic destined to subprefix `10.92.74.0/24` carries 1% of the total traffic of the session, then the adversary will be able to disrupt connectivity to the subprefix by invoking a loss of 1% and likely evading detection. Encryption significantly mitigates this risk by hiding the destination IP address, therefore, making it harder for the adversary to distinguish the victim traffic. Encryption also makes the TCP mechanism *opaque* for the adversary, thus, preventing attacks targetting the TCP mechanism.

3.5 Automated Tunnel Establishment

In this section, we design the mechanism for establishing tunnels in an interdomain setting: Consider two remote stub networks, AS X and AS Y, that decide to deploy stealth probing on their border routers to monitor the availability of interdomain paths between them. A suggestive configuration is to deploy a full mesh of sessions from each border router in X (Y) to each border router in Y (X). In order to assemble this configuration, each border router in X (Y) needs, first, to find out the IP addresses of Y's (X's) border routers and, second, to establish security associations with those routers. We address these problems in turn.

Discovering Remote Routers: Network operators that communicate out-of-band and manually configure the remote IP addresses can provide a straightforward solution to this problem. Manual configuration, despite its limitations, can be useful during limited deployment or as a last resort. DNS can provide a more scalable solution. In a DNS-based system for router discovery, each network

willing to accept stealth probing sessions would register an applicable domain name. For example, using an imaginary `stp` top-level domain, ASes X and Y could respectively register domains `asX.stp` and `asY.stp`. Top-level DNS servers would respond to queries for any subdomain of `stp` (such as `asX.stp` and `asY.stp`) with the IP addresses of the subdomain's border routers. In this way, the discovery of remote routers only requires a local configuration of the relevant list of domain names. DNSSEC and replication of top-level DNS servers can protect the authenticity, integrity, and availability of DNS responses that would also benefit the host traffic.

Establishing Security Associations: IPsec uses the Internet Key Exchange [13] to establish Security Associations (SAs). IKE authenticates peers using either preshared secrets or public-key certificates. Preshared secrets are typically exchanged out-of-band and configured manually. Certificate-based authentication scales better but requires an associated certification authority. Contrary to secure routing proposals such as S-BGP [18], stealth probing does not require a globally trusted certification authority. In our example, X and Y can choose to obtain certificates from a mutually trusted authority irrespective of the certificates used in other sessions. Note that failure of IKE to successfully perform a key exchange implies a failure in the corresponding path and, therefore, that this path should be avoided. DoS attacks targetting the key exchange can be prevented by using the Just Fast Keying (JFK) protocol [5] in lieu of IKE.

4 Deployment Incentives

Designing stealth probing as an extension to IPsec offers benefits beyond secure monitoring, providing additional incentives for deployment. In this section, we give examples of benefits reaped by stealth probing in security, traffic engineering, and network troubleshooting.

Security Benefits. IPsec tunnels at the edge routers of a network infrastructure (a) protect insecure host-to-host communications, (b) prevent traffic-analysis attacks that host-to-host encryption does not prevent, for example, by hiding the source and destination addresses of data traffic, (c) render misdirection attacks that divert traffic for eavesdropping and traffic analysis ineffective, (d) enable ISPs to offer value-added services like secure VPNs to customers, (e) mitigate denial-of-service attacks that use source-address spoofing, and (f) mitigate spam. We discuss now how IPsec tunnels help mitigate DoS and spam.

Spoofing has been routinely used in DoS attacks to hide the location of offending machines. Although the availability of massive botnets has made non-spoofing attacks currently more attractive for the attackers, as mitigation strategies for non-spoofing attacks emerge [31] attackers are likely to resume their strategies. IPsec tunnels enable an edge network to filter inbound packets based on their originating network and, therefore, discard packets that are spoofed. Contrary to ingress filtering [10], this technique requires the configuration of filters at the target network that has the incentives to deploy them. Consider, for example, IPsec tunnels between networks X and Y and assume that X hasn't applied ingress filters. If a host in X uses a fake source address belonging to a different prefix, then Y can readily detect the forgery and configure filters to block it. Inbound packets not belonging to an IPsec tunnel could receive a lower priority at the receiving network (for example, they could be scheduled at a lower-priority queue), especially during an attack.

A recent study on how spammers exploit the network [28] reveals that spammers often employ source-address spoofing (in a method called *direct spamming* and prefix hijacks (in a method that exploits the *BGP spectrum agility*). As discussed above, IPsec tunnels can filter spoofed packets. IPsec tunnels can also protect against spammers doing hijacks as the spammer would not be able to receive the authentication credentials of the hijacked victim network. Therefore, he wouldn't be able to to establish tunnels with the destination networks of the victim, resulting in spam getting discarded at the border routers of those networks.

Traffic Engineering Benefits. IP tunnels (that encapsulate ESP packets) enable edge networks to engineer their interdomain traffic. Consider, for example, the problem of controlling inbound traffic at a multihomed stub network. Existing techniques are based either on a combination of NAT and DNS that provide limited resilience to failures or on BGP hacks (such as *AS path prepending*) that provides better resilience at the expense of coarse control of the inbound traffic. Using the IP tunnels terminating at their borders routers, edge networks can dynamically negotiate bilaterally with source networks the border router to receive inbound traffic and, therefore, achieve traffic control in a resilient fashion and at a fine granularity.

Network Troubleshooting Benefits. Stealth probing enforces fate sharing between data traffic and probes, which is broadly useful for troubleshooting network problems. For example, simple ICMP echo requests and replies may be treated differently from data packets either because of MTU size limits or packet filters that discard traffic based on the protocol or port numbers. Stealth probing avoids this problem as the probes are, in fact, regular data packets.

5 Implementation

We prototyped stealth probing in software. The software implementation is not a mere artifact of the prototyping effort. Open-source routers serve as research plat-

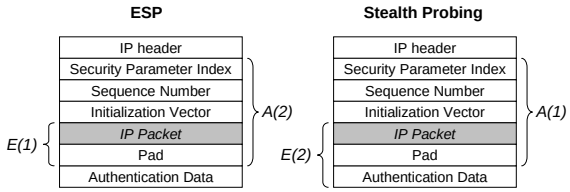


Figure 3: Packet format of ESP in tunnel-mode. E stands for encryption and A for authentication. ESP performs encryption before authentication (left). Stealth probing performs the reverse (right).

forms [4] and are also offered as commercial products, typically deployed at enterprise networks. We used the Click [19] framework for building software routers. By comparison to the native Linux support for routing, Click has better support for extensibility. This enabled us to efficiently implement data forwarding functionality critical to the performance of our system (such as the integration of IPsec security policy with data forwarding outlined in Section 5.3) in a straightforward fashion.

A Click router is a set of interconnected packet processing modules running in a kernel thread. These modules, also called *elements*, are C++ objects. Elements and their interconnections can be represented by a directed graph, also called a Click *configuration*, that characterizes the processing flow of packets. Stealth probing was implemented using modules that extend an existing IP router Click configuration. In the rest of this section, we present how we implemented stealth probing and also our implementation choices.

5.1 Concealment and Integrity Assurance

The encryption and authentication required for concealment and integrity assurance are implemented using the Encapsulating Security Payload (ESP) protocol [17] of IPsec in *tunnel mode*, which provides end-to-end cryptographic protection at the IP layer. In tunnel-mode ESP, IP packets are encapsulated by an ESP header and trailer containing cipher-specific fields. The ESP packet is tunneled by encapsulating it in a new IP header (see Fig. 3). Standard symmetric ciphers can be used to protect the packet. ESP uses a 32-bit sequence number as part of a mechanism that protects from replay attacks. This mechanism is also used in stealth probing to protect from attacks that reorder packets.

Our implementation of ESP is fully compliant with the ESP standard with the exception that authentication is performed before encryption (see Section 5.2). We used the AES encryption cipher and the HMAC-SHA1 authentication cipher; code for those ciphers was obtained from the OpenSSL toolkit.

5.2 Sampling Based on the Keyed Hash

Several possibilities exist for implementing the sampling process. One possibility is to hash immutable parts of the inner IP packet (before encryption at ingress and after decryption at egress) using, for example, modular division as in trajectory sampling [9]. Collisions of the hash function resulting from identical inbound packets can be prevented by also feeding the sequence number of the ESP packet as input to the hash. In order to avoid the additional overhead from modular operations, we chose a different implementation of the sampling process that leverages the pseudorandom output of the keyed hash that authenticates the packet. Using the authenticator, sampling can be performed by comparing its value (two bytes in our implementation) to a threshold. However, ESP sends the authenticator in the clear (see Fig. 3), making it easy for the adversary to distinguish the probes. We address this by inverting the order of encryption and authentication (see Fig. 3). If encryption is performed in Cipher Block Chaining (CBC) mode, performing authentication before encryption has been shown to be secure [20]. Because SHA1 behaves like a random function, the sampling probability is the same for each packet, resulting in an unbiased selection of probes.

5.3 Integrating Security Policy with Forwarding

IPsec policy specifies what traffic should be protected and how. IPsec defines a Security Policy Database (SPD) to store this information at the ingress router through which policy is enforced using packet filters. Rather than implement a separate SPD, we integrate its functionality into the Forwarding Information Base (FIB) of the router in a fashion that may obviate the need for additional filters. The FIB stores the outbound interface per destination prefix and must be looked up for inbound packets to make a forwarding decision. We extend the FIB to support SPD functionality by adding to each FIB entry an additional field. This field stores a pointer either to the IPsec SA for the corresponding destination prefix or to NULL, if stealth probing has not been configured for that prefix. Configuring stealth probing sessions at a granularity finer than the existing prefixes in the FIB can be achieved by installing FIB entries for more specific prefixes (as the *longest prefix match rule* selects the most specific entries). Access Control Lists that match the five-tuple can provide even finer control.

Encapsulation at the ingress router and decapsulation at the egress router should be performed using the same SA. However, the egress router must decapsulate the packet before accessing the FIB, making the abovementioned technique used by the ingress router inapplicable.

In order to retrieve the SA, the egress router maintains a Security Association Database (SADB) accessed using the Security Parameter Index (SPI), a value that is stored in the ESP header of the incoming packet. The SPI is a 32-bit integer that, together with the address of the egress router, uniquely identifies the SA.

Scalability: Using the FIB to enforce IPsec security policy has the advantage that, irrespective of the number of outbound IPsec tunnels, the security association for encapsulating a packet is retrieved simply by following the pointer of the FIB entry. Furthermore, the lookup delay introduced by the consistent hashing operation, if more than one IPsec tunnels are configured for the corresponding destination prefix, grows logarithmically with the number of tunnels and, therefore, is constant for all practical purposes. The only parameter that can affect the SA lookup delay, and, thus, the scalability of the design, is the granularity of security policy. Adding more prefixes in the FIB when subprefixes require separate SAs will increase the FIB lookup delay. Installing access lists will increase the packet processing delay. Noticeable delays of those kinds could only arise in a widespread deployment, in which case operational practice would likely suggest a preferable method for implementing granular policies. The number of inbound tunnels at an egress router only affects the size of the hash table that stores the SAs.

5.4 Consistent Traffic Splitting

Splitting inbound traffic to multiple outbound paths is usually performed using a hash function such as CRC16: The range of the hash function is divided into intervals, where each interval corresponds to an outbound path, and hashing maps inbound packets to the intervals. Most routers hash the source and destination IP address fields of the packet; hashing the five-tuple is also supported by some routers. In the general case, traffic is split into unequal proportions (decided by a load balancing algorithm such as [11, 15]). We consider here the simpler case of an even traffic distribution implying equal-length intervals. This case may arise in practice if stealth probing makes a binary decision to either use or avoid each path.

Even for this simple case, the abovementioned traffic-splitting technique is known to perform poorly when paths are added or deleted dynamically [14]. The reason is that additions and deletions typically result in significant unnecessary changes to the mapping between paths and TCP flows that may disrupt the latter. Consistent hashing [16] can minimize changes when objects (such as web pages) are mapped to a dynamic set of bins (such as web caches). We have implemented a traffic splitting scheme for mapping inbound IP traffic to outbound paths based on consistent hashing that we evaluate in the next section. To our knowledge, this is the first use of consistent hashing as an IP data forwarding module.

5.5 Performance Monitoring

Upon sampling a packet, the egress router responds by sending an ACK to the ingress router. To enable the ingress router to match the ACK with the corresponding probe, the ACK includes the SPI of the tunnel and the 32-bit sequence number of the probe. We also authenticate the ACK using the SA of the probe, which prevents (on-path or off-path) adversaries from forging it. Receiving and generating ACKs is handled by the corresponding Click kernel threads at the ingress and egress routers, respectively, to avoid the overhead of using a separate process.

Timing by the Kernel: Measuring losses and round-trip delays requires timing the probes and ACKs at the ingress router. Keeping a timeout per probe would meet the timing requirements of the measurements and at the same time make the result immediately available for monitoring performance. However, this method would incur significant processing overhead for scheduling and cancelling timeouts and because of the frequent interruption of the kernel thread. We implemented a different timing method that can decrease the CPU load by adding configurable delays to the availability of measurement results. According to this method, the data forwarding modules are only responsible for logging and timestamping probes and ACKs while losses and round-trip delays are computed by reading the logs. Logs are read by a Path-Performance Monitor (PPM) that has been implemented in user space. We elaborate on the PPM later in this section.

The interface between the kernel and PPM is based on a *circular buffer* that essentially acts as a queue between them (see Fig. 4). The circular buffer is exported as a regular file using a Linux proc-like filesystem called *clickfs*. There are two circular buffers, one for the probes and one for the ACKs. An applicable size for each buffer can be determined by the sampling rate, the packet rate, and the round-trip time. For example, if the bit rate of incoming traffic is 1Gbps and the average packet size is 500B, then the packet rate is 250000pps. Assuming also a sampling rate of 2% and a round-trip time of 1sec, the log should contain about 5000 entries. Each log entry contains the 32-bit SPI, the 32-bit sequence number, and a 64-bit timestamp. Therefore, 5000 entries correspond to 80KB of memory.

Monitoring in User Space: The Path-Performance Monitor (PPM) is a daemon implemented in Java that reads the probe- and ACK-logs in order to measure losses and round-trip delays. PPM should be invoked frequently enough to prevent *rollover* of the circular buffers

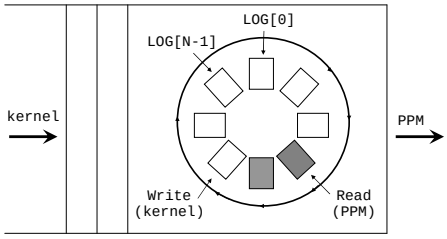


Figure 4: Circular buffer acting as a queue between the kernel and the Performance Monitor.

that would imply loss of measurement data. Rollover could be prevented by forced context switches from the kernel to PPM that we decided against because they are intrusive. The kernel adjusts instead the priority of the PPM daemon according to the occupancy levels of the buffers and warns PPM after a rollover.

PPM processes the logs using the following algorithm: First, the entries of the probe-log are added to a hash table. Then, for each entry in the ACK-log the hash table is queried for a matching probe. If a probe is found, the round-trip delay is calculated and the probe is removed from the hash table. Otherwise, the ACK is considered spurious and it is silently discarded. If all entries in the ACK-log have been processed and there are remaining probes in the hash table, the probe timestamps are compared to the current time for detecting possible losses.

5.6 Summary of Control Flow

In this section, we outline the packet processing steps at the ingress and egress routers of a stealth probing session. For simplicity, we omit those steps involved in processing ACKs. The Click configuration, i.e., the directed graph of processing elements and their interconnection according to how packets flow between elements, is shown in Fig. 5.

Processing Path at Ingress: A packet received from an input port is first classified. Following classification and assuming that the protocol number corresponds to IP, the Ethernet header is removed and the checksum on the IP header is computed (element *Check-IP-header*). *IPsec-Lookup* determines whether a packet belongs to a stealth probing session if so, to which session to direct the packet. *IPsec-Lookup* either retrieves the corresponding SA from the FIB or, assuming that stealth probing is not configured for the packet's destination address, determines the output port and directs the packet for standard IP processing (decrement TTL, compute checksum, etc.). If an SA is found, the ESP header is constructed (*ESP-Encapsulation*), the HMAC is computed (*Authentication-HMAC*), and sampling is performed using the first two bytes of the HMAC (*Sampling*). If a

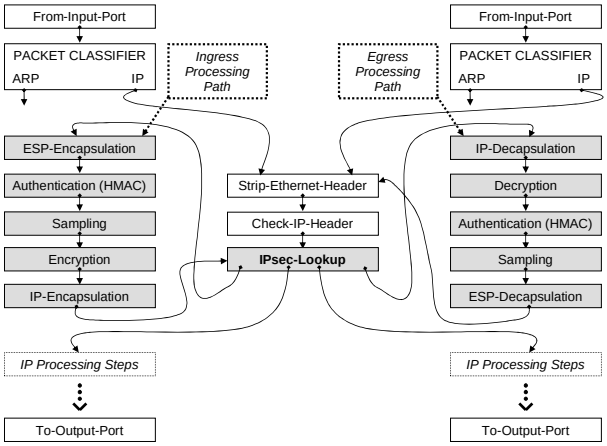


Figure 5: Click configuration of stealth probing.

decision is made to sample the packet, a corresponding entry is added to the log. After the packet is encrypted and encapsulated in a new IP header in module *IP-Encapsulation* (with source address of the ingress router and destination address of the egress router), *IPsec-Lookup* determines the outbound interface for forwarding the packet and directs the packet for the standard IP processing steps.

Processing Path at Egress: At the egress router, the determination of whether a packet belongs to a stealth probing session and, if so, which session should process the packet is also made at the *IPsec-Lookup* element, which either looks up the FIB or the SADB depending on the packet's destination address and protocol number. Assuming that the destination address of the packet is an address of this router and the protocol number corresponds to ESP, then SADB uses the SPI to retrieve the SA. Following that, the outer IP header is removed (*IP-Decapsulation*), the packet is decrypted (*Decryption*), authenticated (*Authentication-HMAC*), and sampled (*Sampling*). If sampling selects the packet, an ACK is generated and authenticated using the same SA as the incoming packet. Then the ESP header is removed (*ESP-Decapsulation*), and *IPsec-Lookup* is invoked for a second time to find the outbound interface for forwarding the inner IP packet and direct it for standard IP processing.

6 Evaluation

In this section, we evaluate our prototype implementation, emphasizing on packet-forwarding efficiency. The main purposes of the evaluation are, first, to determine the latency that stealth probing adds on top of IPsec and destination-based forwarding, second, to compare how the throughput of stealth probing compares to

the throughput of IPsec and destination-based forwarding, and, third, to determine the CPU resources that stealth probing requires for path performance monitoring. We also evaluate the impact of consistent hashing on throughput.

We present our experimental methodology in Section 6.1. Section 6.2 evaluates the processing delay that stealth probing adds to data packets. We evaluate TCP throughput in Section 6.3 and the impact of path-performance monitoring on the system in Section 6.4. In this section, we also identify an inefficiency of the Click implementation regarding packet encapsulation. We show in Section 6.5 that, despite this inefficiency, stealth probing performs comparably to destination-based forwarding in a realistic traffic mix.

6.1 Experimental Methodology

We measure latency using a Click profiling tool that counts and aggregates the cycles each processing module consumes. We evaluate throughput by measuring the maximum data throughput, defined as the Bulk Transfer Capacity (BTC), and the maximum packet throughput, defined as the maximum loss-free forwarding rate when input traffic satisfies a given distribution. BTC is the achievable throughput of a bulk transfer TCP connection [27], which is the maximum throughput that can be attained in practice. We measure BTC using Iperf 2.0.2, maximum packet throughput using custom software, and CPU loads using the *top* system utility.

We set up a testbed of four commodity PCs connected in series by three Ethernet links ($H_1 \rightleftharpoons R_1 \rightleftharpoons R_2 \rightleftharpoons H_2$). R_1 and R_2 run Click 1.5.0 and serve as routers. H_1 and H_2 serve as hosts. We set the MTU on links $H_1 \rightharpoonup R_1$ and $R_2 \leftharpoondown H_2$ to 1400-bytes to avoid fragmentation because of IP and ESP encapsulation. Propagation delay is negligible and the testbed is isolated from cross traffic.

Each link consists of crossover cables connected to dedicated Ethernet interfaces. Each PC has a 2.8 GHz Pentium-4 processor with 1 GB of RAM and running a Linux 2.6.16 kernel. We selectively used both 100Mbit (Realtek) and 1Gbit (D-Link) Ethernet cards in our experiments. The Ethernet device drivers used in our setup do not support "device polling," a technique used by Click for improving performance. The effects are noticeable in the 1 Gbit interfaces. Device polling would improve throughput performance of all configurations that we measure and, therefore, it does not affect our comparisons. Note also that despite the absence of physical layer contention at the links, routers cannot send and receive packets simultaneously as they are limited by the DMA (Direct Memory Access) chip, which can process at most one packet at a time. This negatively affects the throughput performance of stealth probing, more than the other protocols we compare, because stealth probing invokes additional traffic (i.e., ACKs).

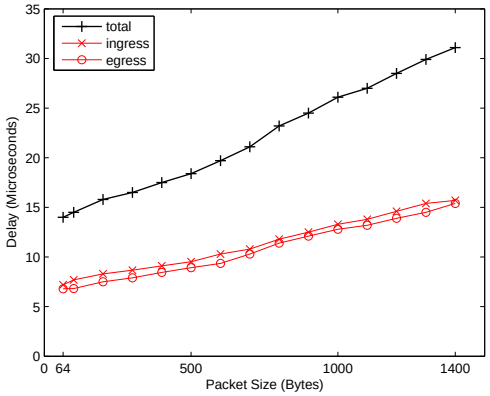


Figure 6: Processing delay that stealth probing adds to a non-probe packet as a function of the packet size. Probes are delayed 10.7 μsec more, irrespective of their size.

6.2 Processing Delay

Figure 6 shows the processing delay that stealth probing modules at the ingress and egress routers add to a non-probe packet as a function of the packet size. We measured this delay using the Click profiler to record the cycle counts of 50 echo requests sent from H_1 to H_2 going through the tunnel from R_1 to R_2. We observed a small variability in the measurements on the order of 0.1 μsec due to interruptions of the Click kernel thread by the scheduler. We report the the minimum cycle counts at ingress and egress and their sum. Stealth probing adds a total processing delay that varies between 14 and 31 μsec, depending on the packet size. The increase in delay for larger packet sizes is due to additional processing required by the authentication and encryption ciphers. If a packet is selected to serve as a probe, there is an additional delay of 2.5 μsec at ingress (to update the log) and 8.2 μsec at egress (to generate the ACK), for a total of 10.7 μsec, independent of packet size. By comparison to ESP, non-probes only require an additional comparison operation for the sampling decision; probes are more expensive (by 10.7 μsec) but their delay is only invoked $x\%$ of the time, where x is the sampling rate.

6.3 Data Throughput

In this section, we measure the effect of stealth probing on data throughput by configuring an Iperf client on H_1 and an Iperf server on H_2. Each experiment has a duration of 20 sec and is repeated ten times; we report the average. Fig. 7 shows data throughput of four configura-

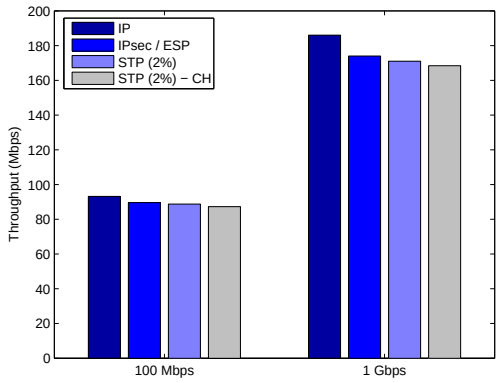


Figure 7: Data throughput with 100Mbps and 1Gbps links.

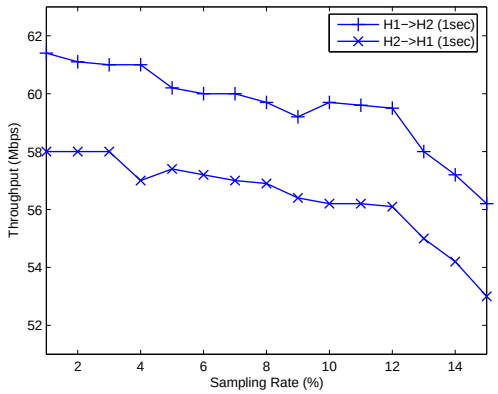


Figure 8: Impact of sampling rate on TCP throughput.

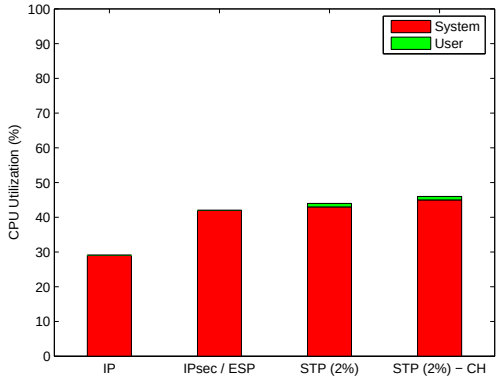
tions between R_1 and R_2: IP, IPsec (ESP), stealth probing at 2% sampling rate, and stealth probing (at 2%) with consistent hashing (selecting one among 100 tunnels). We repeat the measurements for 100 Mbps and 1 Gbps links. In the first case (100 Mbps links), the stealth probing data rate is 5% less than IP and 1% less than ESP. The overhead of consistent hashing decreases the data rate by 2%. In the second case (1 Gbps links), the stealth probing data rate is 8% less than IP and 2% less than ESP. Consistent hashing decreases the data rate by 2%. Stealth probing at zero sampling rate had approximately the same performance as ESP.

Impact of Sampling Rate: To measure the impact of the sampling rate on data throughput, we create additional traffic to compete with the stealth-probing ACKs by adding a TCP session in the opposite direction using an extra Iperf client and server. In the resulting configuration there are two TCP sessions one in the direction $H_1 \rightarrow H_2$ and the other in the direction $H_1 \leftarrow H_2$. Both TCP connections start simultaneously in each experiment, have a duration of 20 sec and are repeated ten times; we report the average of the ten measurements. Fig. 8 shows the data throughput of the two TCP sessions as the sampling rate varies from 1% to 15%; link bandwidth is 100 Mbps. We observe that up to a 10% sampling rate, throughput decreases slowly as the sampling rate increases. The decrease in throughput is slow despite the linear increase in the ACK-traffic and additional processing overhead for generating and reading logs. We attribute this to the small size of stealth-probing ACKs (80-bytes in total) and the efficiency of performance monitoring. We elaborate on performance monitoring in the following section.

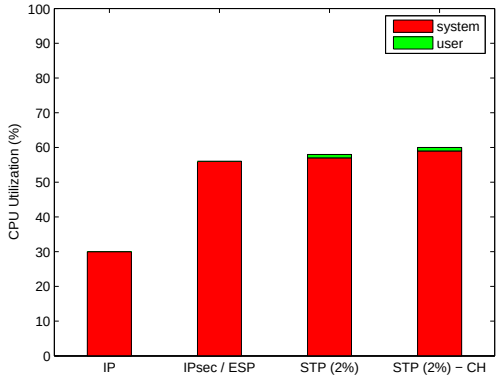
6.4 CPU Resources

Figures 9(a) and 9(b) show CPU utilization during the data-throughput experiments of the previous section. By comparison to IP forwarding, stealth probing increases the CPU load 15% in the case of 100 Mbps links and 28% in the case of 1 Gbps links. Consistent hashing increases the CPU load by 1% in both cases. We may observe in the figures that most of the CPU resources are consumed in packet forwarding. In actual deployment, packet forwarding would typically be assigned to dedicated hardware. Also shown in the figures is that the user-space PPM module contributes only 1% of the load in both cases. This experiment, therefore, demonstrates that PPM is non-intrusive for the processor. We, thus, believe that performance monitoring could be the responsibility of the CPU of high-end routers, implementing stealth probing in the data plane, without interfering with routing-protocol processing.

Fig. 9 shows that the impact of stealth probing on the CPU is significant. We, therefore, decided to investigate the source of this inefficiency. In the course of the investigation, we identified an inefficiency of Click in the standard procedure it uses for encapsulating packets. Fig. 10 shows the CPU utilization of IP and of IP with a *dummy encapsulation and decapsulation step* inside a random processing module, using the same experimental setup as Section 6.3. This dummy step alone increases the CPU load by 5%, in the case of 100 Mbps links, and by 10.5% in the case of 1 Gbps links. This experiment, therefore, reveals a weakness in the Click software, likely attributed to inefficient memory management. We further discuss this inefficiency in the next section.



(a) 100 Mbps



(b) 1 Gbps

Figure 9: CPU Utilization of IP, IPsec, stealth probing sampling at 2%, and stealth probing with consistent hashing in the data throughput experiment.

6.5 Packet Throughput

In this section, we compare the maximum zero-loss packet throughput of stealth probing with that of destination-based forwarding. The comparison is inevitably affected by the aforementioned inefficiency of Click. Fig. 11(a) shows the maximum zero-loss packet throughput of IP, of IP with the dummy encapsulation and decapsulation step, of stealth probing at zero sampling rate, and of stealth probing (at zero rate) with consistent hashing (selecting one among one-hundred tunnels) under a realistic traffic-mix. In the traffic mix, data packet sizes are drawn from a probability distribution that approximates traffic collected by NLANR during February, 2001 [2, 3]. We also show the breakdown of performance using fixed-size input traffic. Note that the dummy encapsulation step alone decreases throughput by approximately $20,000$ packets. Stealth probing

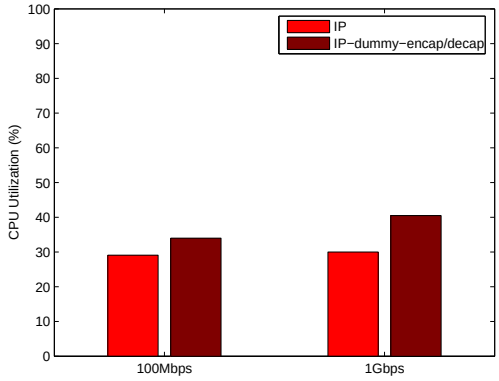


Figure 10: CPU utilization of IP and of IP with dummy encapsulation and decapsulation.

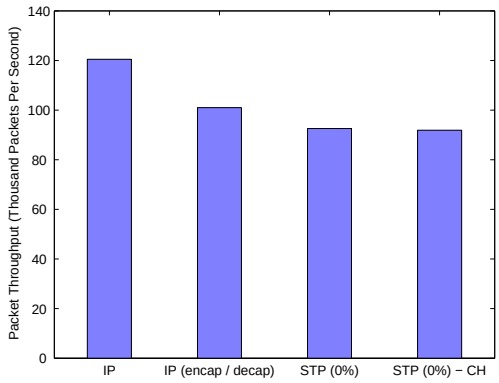
can sustain a zero-loss packet rate that is 23.1% lower than IP despite the encapsulation inefficiency. Adding consistent hashing to stealth probing decreases the zero-loss packet rate by 0.8%. Fig. 11(b) shows the breakdown of performance for fixed-size packets. We observe that the encapsulation inefficiency has a smaller impact as the packet size increases becoming negligible at 1400-byte packets used in the data-throughput experiments of Section 6.3. We are currently investigating the actual source of the inefficiency and possible ways to fix it.

7 Related Work

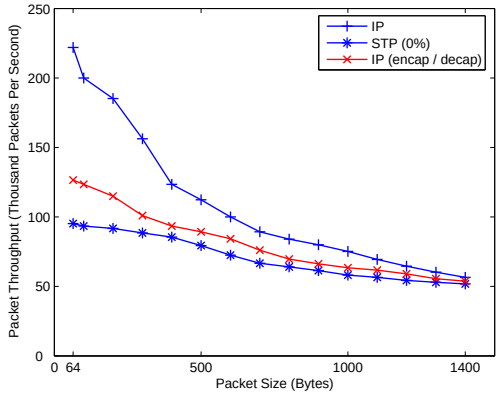
Perlman [26] proposed encryption between neighboring routers to make data and control traffic indistinguishable and per-packet acknowledgments to monitor the link between the routers. Stealth probing uses encryption between remote routers to monitor the availability of paths and relies on implicit probing based on sampling, which makes the network overhead unobtrusive.

Fatih [24] is a system for detecting and isolating malicious routers using a traffic validation method between terminal routers that relies on clock synchronization. Therefore, successful clock-synchronization attacks could enable successful attacks against availability. Because Fatih does not specify a secure clock synchronization protocol to be used with the system, we cannot directly compare its security and efficiency. Stealth probing does not depend on clock synchronization. Furthermore, Fatih sends data packets in the clear, allowing an adversary to selectively target the traffic. Stealth probing tunnels and encrypts the traffic to prevent this attack.

Secure traceroute [25] is a scheme for secure localization of faulty links that could conceivably be applied at the path level. In secure traceroute, data packets between an *initiating* router and a *responding* router are selected



(a) Mixed Input Traffic



(b) Fixed Packet Size

Figure 11: Packet throughput (a) under mixed traffic and (b) under fixed-size traffic.

by the initiator to serve as probes by embedding in them secret identifiers. However, the responder stores replies for later retrieval [22], opening the possibility for delay attacks. Furthermore, data packets are neither encrypted nor tunneled, allowing an adversary to target individual components of the aggregate traffic.

Listen [29] and the data-plane monitor of the Feedback-Based Routing system [34] detect data-plane attacks by a combination of passive measurements of TCP traffic and insecure active probing. As such they have the limitations outlined in Section 2.2.

We previously introduced the idea of stealth probing in a short paper [6]. This paper makes the following additional contributions: First, we articulate in detail the design decisions such as why we choose to monitor availability from edge routers instead of hosts. Second, we refine the design and, for example, dismiss secure active probing, proposing instead to split traffic on multiple paths and do passive probing on each path. Third, we present a methodology for infering path-performance using statistical sampling theory. Fourth, we present in detail a prototype implementation of the system and, finally, a thorough evaluation of the prototype in a testbed.

8 Conclusion

Resilience has always been an important priority in the design of IP networks [8]. However, the threat model has changed significantly in recent years with increasing attacks against the Internet infrastructure by subverted systems. Despite these threats, IP networks still rely on routing protocols that treat control-plane messages as an accurate indication of whether and how the data plane delivers user traffic. Periodic beaconing between adjacent routers cannot detect whether routers are maliciously discarding or redirecting data packets, while continuing to forward control messages and active probes to evade detection. Instead, we argue for secure, passive, path-level probing between edge routers to identify availability problems. In stealth probing, a fraction of the data packets serve as implicit probes and trigger acknowledgment messages that enable accurate estimates of packet loss and round-trip delay, even if an adversary lies in the data plane along the path. Experiments with our prototype implementation demonstrate that stealth probing can operate at high speeds.

Stealth probing can be readily deployed in today's Internet for three important reasons. First, IP tunnels (that encapsulate ESP packets) can be deployed accross legacy routers and ASes. Therefore, stealth probing is *backward compatible* with the existing infrastructure. Edge networks using secure VPNs would not need to upgrade their routers. Furthermore, commercial routers increasingly offer IP tunneling and encryption at line rates. Second, IPsec security associations can be established bilaterally, making stealth probing *incrementally deployable*. Any pair of ASes can deploy stealth probing irrespective of the participation of other ASes. As such, stealth probing offers immediate benefits even during limited deployment. Finally, stealth probing is *incentive compatible* offering additional benefits from mitigating DoS attacks and spam to engineering interdomain traffic.

Our work on stealth probing is part of a larger vision for how to design a secure routing system for the Internet. We believe that secure routing protocols, while useful, are neither sufficient nor necessary. A secure control plane cannot easily defend against colluding adversaries or ensure that the data packets actually follow the advertised path. In addition, a secure interdomain routing protocol relies on having accurate registries and a public-key infrastructure, as well as having the routers perform cryptographic operations on the control messages. Instead, we argue that the interdomain

routing system should be designed to provide *availability*—ensuring that an edge network has at least one path that reaches the intended destination—along with secure path-level probing to identify faulty paths. We believe that stealth probing, combined with a multi-path interdomain routing protocol, can lead to a robust, secure Internet infrastructure for the future.

References

[1] *Cisco Systems: Optimized Edge Routing (OER)*.

[2] *http://advanced.comms.agilent.com/*.

[3] *http://pma.nlanr.net/Datacube/*.

[4] *http://www.xorp.org/*.

[5] W. Aiello, S. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. Keromytis, and O. Reingold. Just Fast Keying: Key agreement in a hostile Internet. *ACM Trans. Information and System Security*, 7(2):242–273, May 2004.

[6] I. Avramopoulos and J. Rexford. Stealth probing: Efficient data-plane security for IP routing. In *Proc. USENIX Annual Technical Conference*, May/Jun. 2006.

[7] F. Bacceli, S. Machiraju, D. Veitch, and J. Bolot. The role of PASTA in network measurement. In *Proc. ACM SIGCOMM*, Sept. 2006.

[8] D. Clark. The design philosophy of the DARPA Internet protocols. In *Proc. ACM SIGCOMM*, Aug. 1988.

[9] N. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Trans. Networking*, 9(3):280–292, Jun. 2001.

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

[11] D. Goldenberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. In *Proc. ACM SIGCOMM*, Aug./Sept. 2004.

[12] F. Guo, J. Chen, W. Li, and T. Cker. Experiences in building a multihoming load balancing system. In *Proc. IEEE Infocom*, Mar. 2004.

[13] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, IETF, Nov. 1998.

[14] C. Hopps. Analysis of an equal-cost multi-path algorithm. RFC 2992, IETF, Nov. 2000.

[15] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proc. ACM SIGCOMM*, Aug. 2005.

[16] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *Proc. ACM STOC*, May 1997.

[17] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). RFC 2406, IETF, Nov. 1998.

[18] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (Secure-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, Apr. 2000.

[19] E. Kohler, R. Morris, B. Chen, J. Janotti, and F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, Aug. 2000.

[20] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). In *Proc. CRYPTO*, Aug. 2001.

[21] E. L. Lehmann. *Testing Statistical Hypotheses*. Wiley, New York, 1959.

[22] G. Mathur, V. Padmanabhan, and D. Simon. Securing routing in open networks using secure traceroute. Technical Report MSR-TR-2004-66, Microsoft Research, Jul. 2004.

[23] D. Mcpherson and C. Labovitz. Worldwide infrastructure security report, Volume II. Technical report, Arbor Networks, Sept. 2006.

[24] A. Mizrak, Y.-C. Cheng, K. Marzullo, and S. Savage. Fatih: Detecting and isolating malicious routers. In *Proc. International Conference on Dependable Systems and Networks*, Jun. 2005.

[25] V. Padmanabhan and D. Simon. Secure traceroute to detect faulty or malicious routing. In *Proc. ACM SIGCOMM HotNets Workshop*, Oct. 2002.

[26] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, Massachusetts Institute of Technology, Aug. 1988.

[27] R. Prasad, C. Dovrolis, M. Murray, and KC Claffy. Bandwidth estimation: Metrics, measurement techniques, and tools. *IEEE Network*, Nov.-Dec. 2003.

[28] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proc. ACM SIGCOMM*, Sept. 2006.

[29] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz. Listen and Whisper: Security mechanisms for BGP. In *Proc. Symposium on Networked System Design and Implementation*, Mar. 2004.

[30] S. Thompson. *Sampling*. Wiley Series in Probability and Statistics. Wiley, second edition, 2002.

[31] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker. Ddos defense by offense. In *Proc. ACM SIGCOMM*, Sept. 2006.

[32] D. Wendlandt, I. Avramopoulos, D. Andersen, and J. Rexford. Don't secure routing protocols, secure data delivery. CMU-CS-06-154, School of Computer Science, CMU, Sept. 2006.

[33] W. Xu and J. Rexford. MIRO: Multi-path interdomain routing. In *Proc. ACM SIGCOMM*, Sept. 2006.

[34] D. Zhu, M. Gritter, and D. Cheriton. Feedback based routing. In *Proc. ACM SIGCOMM HotNets Workshop*, Oct. 2002.