Single packet IP traceback in AS-level partial deployment scenario

Turgay Korkmaz*

Department of Computer Science,
The University of Texas at San Antonio,
6900 North Loop 1604 West, San Antonio, TX 78249, USA
E-mail: korkmaz@cs.utsa.edu
*Corresponding author

Chao Gong and Kamil Sarac

Department of Computer Science,
The University of Texas at Dallas,
2601 N Floyd Road, Richardson, TX 75080, USA
E-mail: cxg010700@utdallas.edu
E-mail: ksarac@utdallas.edu

Sandra G. Dykes

Communications and Embedded Systems Department,
Southwest Research Institute,
6220 Culebra Road, San Antonio, TX 78238, USA
E-mail: sdykes@swri.org

Abstract: Denial-of-Service (DoS) attacks commonly use IP spoofing to hide the identity and the location of the attack origin. To defend against various DoS attacks and make the attacker accountable, it is necessary to trace IP packets regardless of their source addresses. In this direction, log-based IP traceback is a promising and powerful approach due to its ability to traceback even a single packet. However, the global deployment of log-based IP traceback at all the routers in the internet requires a significant amount of modifications in the routers and introduces a serious operation and management overhead. To facilitate global deployment, we consider the Autonomous Systems (AS) level deployment of log-based IP traceback and accordingly propose a new mechanism called AS-level Single Packet Traceback (AS-SPT). We then evaluate the performance and overhead of the proposed AS-SPT under various partial deployment scenarios.

Keywords: IP traceback; single packet IP traceback; Denial-of-Service (DoS) defense.

Reference to this paper should be made as follows: Korkmaz, T., Gong, C., Sarac, K. and Dykes, S.G. (XXXX) 'Single packet IP traceback in AS-level partial deployment scenario', *Int. J. Security and Networks*, Vol. X, No. Y, pp.XXX–XXX.

Biographical notes: Turgay Korkmaz is an Assistant Professor in The University of Texas at San Antonio, Texas, USA. He received his PhD from The University of Arizona in December 2001. His research interests include QoS-based routing, multiple constrained path selection, network measurements, and network security.

Chao Gong is a PhD student in the Department of Computer Science, The University of Texas at Dallas. His research interests are in network management and security.

Kamil Sarac is an Assistant Professor at The University of Texas at Dallas, Richardson, Texas, USA. He received his PhD from the University of California Santa Barbara in July 2002. His research interests include network monitoring and measurements, overlay networks, network security, group communication and IP multicast.

Sandra G. Dykes is a Senior Research Scientist at Southwest Research Institute in San Antonio, Texas, USA. She received her PhD from The University of Texas at San Antonio in December 2000. Her research interests include cooperative internet protocols, network security, and network performance evaluation.

1 Introduction

Detecting the actual sources of Denial-of-Service (DoS) attacks is necessary in both defending against such attacks and making the attackers accountable. However, this is a challenging task because of *IP spoofing*, where a wily attacker places an arbitrary IP address as its own IP address. In other words, when an attack packet is received, it is difficult to know who is the actual sender. As a results, tracing an IP packet regardless of its source address is an important utility in defending against various DoS attacks that are severely threatening the utility of the internet (Garber, 2000; Moore et al., 2001).

Tracing the path of an IP packet back to its origin is known as *IP traceback* and has been extensively investigated during the last several years. As we discuss in Section 2, the existing approaches can broadly be classified as *packet logging* and *packet marking*. In this paper, we focus on the log-based IP traceback approach due to its ability to trace even a single packet. Ability to trace a single packet is necessary particularly in locating an attacker that uses a small number of packets to launch an attack. However, tracing a single packet was historically thought to be impractical as it requires all the packets to be logged.

In response to making single packet traceback feasible in practice, Snoeren et al. (2002) developed the Source Path Isolation Engine (SPIE) architecture, where routers record packet *digests* in a space-efficient data structure, namely Bloom filter (Bloom, 1970). The router level deployment of SPIE is desirable to successfully locate the origin of an attack packet. However, updating all the routers creates a barrier for the initial deployment of the service on a large scale. In addition, as with any new network service, the initial deployment phase will introduce difficulties due to the lack of required management tools and sufficient expertise to operate the service.

In order to facilitate the global deployment of the log-based IP traceback service, we first propose to perform the single packet traceback operation at the Autonomous Systems (AS) level. Accordingly, we design an AS-level Single Packet Traceback (AS-SPT) mechanism. In essence, AS-SPT logs packet digests at the border routers of participating ASes, and then traces a given attack packet toward its origin at the AS level. One of the key advantages of the AS-level traceback approach is that it minimises the required level of deployment at the participating ASes. This is desirable because deploying the service at a subset of the routers (i.e. at the border routers only) limits the required hardware/software modifications and the management overhead for the deploying ASes. This naturally reduces the deployment and operation costs of the service. Despite this fact, it is still unrealistic to assume that all ASes begin to support the proposed traceback mechanism in a short period of time. Accordingly, we enhance the AS-SPT mechanism so that it can continue tracing a single packet under partial deployment scenarios. In addition, we identify several practical issues including the impact of packet transformations and false positives under partial deployment and potential security vulnerabilities. We then present several solution approaches to address these issues. Finally, we use simulations and analytical models to evaluate the performance and overhead of the AS-SPT approach under various partial deployment scenarios.

AS-SPT service provides several important advantages. Firstly, AS level deployment eliminates the need for router level deployment of the service at the core of the network (transit ASes). Secondly, it can be used to monitor and visualise the general trend of the worldwide propagation of new computer worms and viruses. Thirdly, figuring out the network path that the attack traffic follows can improve the efficacy of defense measures such as packet filtering as they can be applied further from the victim and closer to the actual source of the attacker. Fourthly, it can be combined with a router level logging-based traceback mechanism (e.g. SPIE) at the attacker's domain. In this case, by utilising AS-SPT service, one can quickly identify the originating AS of an attack packet. Once the originating AS is detected, the existing router level traceback service can be used to pinpoint the attack origin within the AS. Finally, AS-SPT service does not require a specific type of logging mechanism to be used at the router level. Instead, it can work with different logging-based traceback approaches deployed at the router level at different ASes.

The rest of this paper is organised as follows. Section 2 puts our work in the context of related work. Section 3 describes the proposed AS-SPT approach in detail and addresses various practical issues. Section 4 evaluates the performance of the proposed AS-SPT under partial deployment scenarios. Finally, we conclude our work with a brief look at the future work in Section 5.

2 Related work

The existing approaches for IP traceback can be divided into two groups:

1 logging-based IP traceback and

2 marking-based IP traceback.

Since our work takes on logging-based IP traceback, we now briefly present the related work in this group. For more information about the second group, we refer readers to Belenky and Ansari (2003) and Gao and Ansari (2005).

Logging-based IP traceback mainly involves two key steps (Sager, 1998):

1 routers log the packets that they forwarded

2 victims query/examine the logging information maintained at the preceding routers from the victim towards the attacker.

To reduce the required storage space, logging should be done in a space-efficient manner, foe example using a Bloom filter (Bloom, 1970). Snoeren et al. (2002) used this idea and developed a router-level logging-based IP traceback architecture called 'SPIE'. It is shown that the storage overhead is significantly reduced (down to 0.5% of the total link capacity per unit time).

Researchers have also been working on how to improve SPIE by further reducing the storage overhead of packet digests. In this regard, Lee et al. (2004) proposed to digest packet aggregation units (packet flow or source-destination set) instead of the individual packets. Recording the digests of packet aggregation units reduces the digest table storage requirement. In this case, tracing an individual packet is

accomplished by tracing the packet aggregation to which the packet belongs. Li et al. (2004) proposed that routers probabilistically select a small percentage of the packets and record their digests. This method reduces the storage overhead while losing the ability to trace individual packets. Gong and Sarac (2005) presented a hybrid IP traceback approach based on both packet logging and packet marking. This approach has the same single-packet traceback ability as in SPIE, but incurs less storage overhead through utilising packet marking.

The previous work on logging-based IP traceback focuses on router-level traceback within an AS. Our work is on AS-level IP traceback across multiple ASes. Moriarty (2005) proposed an inter-AS communication protocol, called 'Real-time Inter-network Defense' (RID), to facilitate the cooperation of ASes during an AS-level traceback process. Therefore, RID and our work in this paper can complement each other in implementing an AS-level traceback system.

3 As-level single packet traceback

In this section, we describe the proposed AS-SPT approach and explain how it works under partial deployment. In addition, we identify several important practical issues and discuss potential solutions to address them.

3.1 AS-SPT design

In AS-SPT, participating ASes deploy a logging-based traceback service at their border routers. We do not require any specific approach for the implementation of the logging mechanism. However, we assume that SPIE is the one that is used since it is the most popular approach. According to Sanchez et al. (2001), the logging functionality can be deployed on a 'tap box' that is co-located with the border routers or can be implemented on a line card and inserted into the routers. Finally, depending upon the volume of the traffic crossing over the border routers, the packet logs can be stored at the routers or can be transferred to a stable storage device within the domain.

Each AS-SPT-enabled AS maintains an AS Traceback (AST) server. AST servers monitor the operation of the border nodes logging the packets. AST servers also serve as the main contact point for traceback queries coming from local or remote users. Each AST server keeps track of the AST servers in the neighbouring domains (see Section 3.2 for neighbour discovery). When a traceback query arrives, the AST server queries each of its border nodes. If the packet in question transited through this domain, the AST server gets a SEEN reply from both the ingress and the egress border routers that logged the packet. Using the AS level neighbourhood information, the AST server identifies the preceding AS (in the case of point-to-point connection between the ingress router and the neighbour's router) or the preceding ASes (in the case of a shared media connecting the ingress border router with others). At this point, depending on the local policy, the AST server of the current AS may:

- forward the query to the preceding ASes' AST servers (i.e. recursive traceback mode) or
- send a response back to the original querier with the collected information (i.e. iterative traceback mode).

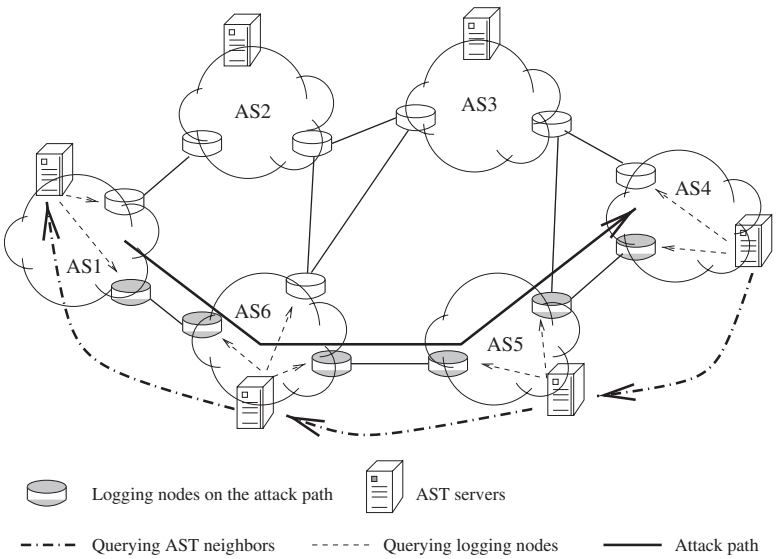
The former case is shown in Figure 1, which illustrates the global view of the AS-SPT approach and the chain of monitors whose logs or digest tables contain the packet in question. In the latter case, the querier contacts the upstream ASes to continue the traceback operation towards the attack origin. Finally, the traceback operation terminates when the last AS is known to be a stub AS or the AST server in that AS can identify the egress border but not the ingress border for the attack packet.

The success of AS-SPT depends on several practical issues including partial deployment of the AS-SPT service, the impact of packet transformations, potential false positives during traceback and security vulnerabilities in the AS-SPT architecture. We discuss these issues below.

3.2 Partial deployment scenario

During a traceback operation, starting from the AS at which the victim resides, the AST server on the attack path can communicate with each other and trace the attack path back

Figure 1 AS-level traceback architecture



to its origin. However, if an AS on the attack path does not support AS-SPT, it interferes with the traceback operation and may potentially halt the process. To work around this problem, we propose two modifications to the basic design presented in the previous subsection.

Firstly, we require participating ASes to keep a list of other participating ASes within a radius R. That is, an AST server will keep the AS-level topology for its R AS-hop neighbours and the addresses of AST servers in the participating ones. As an example, Figure 2 shows the topology information maintained at AS A when $R = 3$. The AST server of AS A will know the addresses of the AST servers at ASes C, F, G, I, J, L, M and N.

Secondly, we propose to *skip* non-deploying ASes if they happen to be on the attack path. For example, a participating AS, say AS A in Figure 2, detects that the AS level path toward the attack origin goes through a non-deploying neighbour AS, say AS B in the example. In that case, AS A skips AS B and forwards the query to the AST servers in the more distant ASes. For example, AS A first contacts the two AS-hop neighbours beyond the non-deploying neighbour, AS G in the example, and then continues to increase the skip level until either it receives a positive response from a contacted AS or it receives negative responses from all the neighbours within the maximum skip radius R. For example, in the above figure, if the attacker is in AS M, no skip is necessary. However, if the attacker is in AS F, AS A needs at least 2-level skip.

Repeated skipping increases the success rate in finding the attacker's AS, but must be carefully controlled to avoid an exponential increase in the number of traceback queries. Each AST server may set a limit on the number of queries forwarded in response to a single traceback request, the query rate (e.g. queries per minute), and the skip level (= 1, 2, ...). In addition, due to the network topology, an AST server may get multiple queries for the same attack packet from a different set of neighbours. In order to reduce the traceback overhead, each AST server remembers the recently served queries and responds to the repetition queries with the information cached locally.

Finally, we discuss issues regarding the collection of the required neighbourhood information. Considering the fact that the AS level topology (i.e. connectivity graph) changes infrequently, we assume that the participating ASes can collect and upload this information to their AST servers in an off-line manner (e.g. using traceroute). Alternatively, the Border Gateway Protocol (BGP) can be utilised to communicate the required information among the neighboring ASes. All AS-SPT deploying ASes advertise the addresses of their AST servers in a BGP attribute to their neighboring ASes. This approach provides an automated way of learning the required information. However, potential AS-level path asymmetry or the existence of secret BGP peering relations may introduce potential inefficiencies.

3.3 Impact of packet transformations

Packet transformations introduce potential difficulties during a traceback operation. In general, several parts of an IP packet are used to generate the packet digest for logging-based traceback purposes. Any modifications on these parts of the packet may result in a different packet digest, making it difficult to continue traceback operation.

IP packets undergo several modifications during their transmission over the wide area network. Firstly, each IP packet has its TTL value decremented and its IP checksum field modified by the forwarding routers. In addition, several other transformations may be applied to IP packets. In order to minimise the impact of packet transformations on the traceback service, we utilise only a subset of the fields in the packet header in computing the log. These fields correspond to the unshaded fields in Figure 3. Considering the most commonly used transformations, for example, packet fragmentation and ICMP error messages, the selection of these fields enable us to proceed with traceback operation in the presence of such transformations.

Although packet transformation is not a common event,[2] a motivated attacker may try to cause such transformations to defeat the traceback mechanism. To avoid such a negative effect, it is necessary to incorporate transformation resistent mechanisms. With this goal in mind, we group packet transformations into three groups and discuss how to deal with the transformed packets in AS-SPT.

In the first group, we consider transformations that occur at the end points of the network, that is, originating subnet and/or destination subnet. These include transformations due to Network Address Translation (NAT) boxes or end-to-end Virtual Private Network (VPN) connections between the two end points. These transformations do not interfere with our mechanism and AS-SPT mainly traces the transformed packet back to its origin AS. Pinpointing the actual attack node may need the transformation information but this

Figure 2 AS level partial deployment scenario

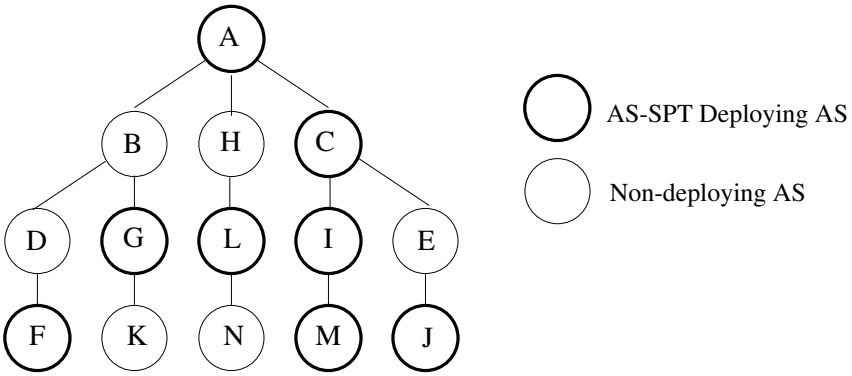
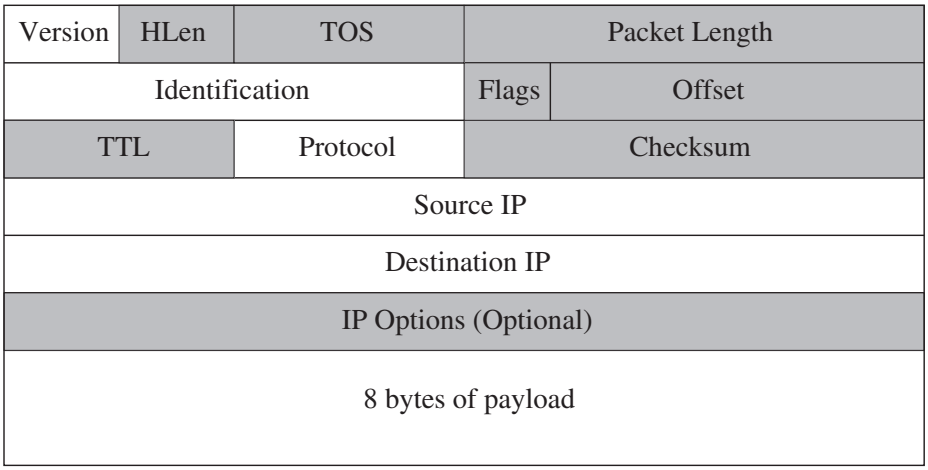


Figure 3 Header fields (unshaded fields) used in the log computation



information can be maintained as part of the router level traceback service mechanism when deployed at the origin AS (Strayer et al., 2003).

In the second group, we consider transformations where the transformed packet includes information about the original packet. These transformations include IP fragmentation and ICMP error messages. In the case of IP fragmentation, for logging purposes, the transformed packet includes the same information as the original packet since we do not consider fragmentation related fields (see Figure 3). In the case of ICMP error messages, these messages include the IP header and the first 8 bytes of the payload of the original packet. This information can then be used to trace the original packet back to its origin AS from the transformation point on.

In the last group, we consider transformations where the original packet undergoes two dual transformations in the network. Typically, these transformations are caused by the existing MPLS, VPN or IP-in-IP tunnels between two intermediate routers on the end-to-end path of the packet. During the first transformation, the packet is encapsulated into another packet/frame and then in the second one the packet is decapsulated into its original form. During a traceback process, the original packet can be traced back to the decapsulation point. Since the packet does not include any information about the transformation, it is not possible to convert the packet back to its transformed version. Therefore, it is not possible to trace the transformed version of the packet in the network.

At this point, one option to proceed with the traceback might be to query up to R AS-hop neighbour ASes of the current AS to search for evidence on the original version of the packet. If the AS level length of the tunnel is shorter than the AS level skip value (i.e. R), we may be able to locate the AS that logged the packet before it went into the transformation. This way, the traceback process may continue until it reaches the origin AS. On the other hand, if the AS level length of the tunnel is longer than the skip level R, then the traceback process returns only a partially detected path, which can still be useful in defense-in-depth efforts.

Finally, the ICMP query mechanism can be used to launch reflector-based flooding attacks. In this context, the attacker spoofs the IP address of the victim and sends an ICMP query message to a third party node, called the 'reflector node'. The reflector then creates and sends an ICMP response packet to the victim site. In these types of attacks, the attacks can only be traced back to the reflector point since the ICMP response message does not include information about the ICMP query message.

3.4 False positives

Bloom filters are flexible data structures and require two configuration parameters:

1 number of hash functions to create packet logs and

2 the required amount of storage space.

Depending on these configuration parameters, each Bloom filter introduces a false positive probability. False positives are undesirable because they may lead to inaccuracies during a traceback operation. Fortunately, false positive rates can be controlled by carefully designing a Bloom filter data structure and by periodically flushing it before it gets saturated during its operation. In the rest of this section, we discuss the mechanisms to control the false positive rates and their potential impact on the success of traceback operations.

Following the discussion in Fan et al. (2000), we can design a Bloom filter with the following properties:

$$m/n = 12,\ k = 8,\ \Rightarrow\ p_{f+} = 0.00314$$

where m is the size of a Bloom filter, k is the size of log information, n is the number of packet digests stored in a Bloom filter, and p_{f+} is the false positive rate. Using this configuration for Bloom filters and assuming that we maintain a separate Bloom filter for each interface of a logging node, for an OC-192 link, the required amount of storage space to log one minute worth of traffic is about 896 MB for an average packet size of 1000 bits. For an OC-3 link, the required amount of storage space is about 14 MB. Considering the fact that AS-SPT utilises Bloom filters only at AS borders, the above figures suggest that achieving an acceptable false positive rate ($p_{f+} \cong 10^{-3}$) is certainly doable with a moderate level of storage requirement at the logging nodes at the borders.

Using the above false positive rate ($p_{f+} = 10^{-3}$) per logging node, the probability of a false positive within an AS level becomes 10^{-6} as both the ingress and egress nodes

need to introduce a false positive within the AS. This figure clearly shows that when we design and operate Bloom filters in an appropriate way, the probability of making a wrong inference during an AS-level traceback operation is extremely small. Using the above false positive rate, the probability of constructing an erroneous AS level attack path of length H AS-hops becomes $p_{f+}^{2H} = 10^{-6H}$, for $H = 1, 2, \ldots$. Finally, even if an AS introduces a false positive and forwards the traceback query to a wrong upstream AS, the upstream AS can consult the AS-level inter-domain routing information to check if the traceback path conflicts with the inter-domain routing policy. That is, when an AS (say AS A) receives an AS-SPT traceback query from a downstream AS (say AS B) for an IP packet destined to a victim X, AS A checks to see if it uses AS B as the next node when forwarding IP packets to victim X. This way AS A may detect and avoid potential inaccuracies caused by false positives during a traceback operation.

False positives during a traceback process may introduce additional query overhead into the process. We now study the impact of false positives on the query overhead. Since the false positive probability is very small, such events will be very rare. As stated by Gorg et al. (2001), rare events lead to simulation run times in the order of months. Instead of using simulations, we attempt to understand the query overhead due to false positives using an analytical model. Specifically, we consider a complete k-ary tree, in which each vertex (AS) has k children (neighbour ASes) and each branch has the same depth denoted by d. In addition to k children, the root is also connected to another AS (say ASx). Suppose that all the ASes deploy AS-SPT and the root of the tree is the victim's AS while ASx is the attacker's AS. If there were no false positives, then the number of queries would be 1 per traceback request, as sent from the root to ASx. However, due to false positives, more queries would be utilised. It is easy to show that the expected number of extra queries per traceback request due to false positives in the above k-ary tree is

$$E[\text{extra query}] = \sum_{i=1}^{d} (kp_{f+})^i$$

From this equation, we can determine the expected number of extra queries as

$$E[\text{extra query}] = \left(\frac{1 - (kp_{f+})^{d+1}}{1 - kp_{f+}} - 1 \right)$$

Figure 4 depicts the relationship between E[extra query] and d under different values of k and p_{f+}.

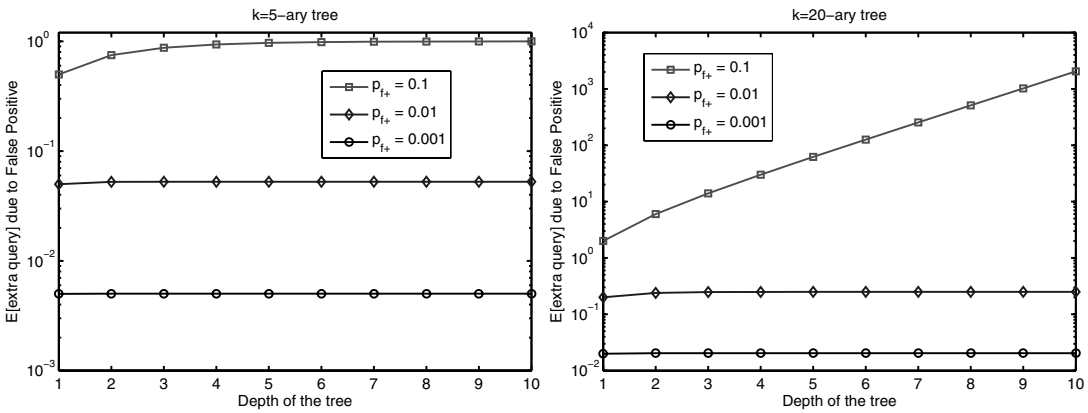
When the false positive rate is within practical/typical values (e.g. in the order of 10^{-3} or 10^{-2}), there is a negligible increase in the query overhead per traceback request. However, if the false positive rate is unreasonably high (e.g. 10^{-1}), then there will be a significant query overhead. This suggests that a Bloom filter with a target false positive rate of $p_{f+} = 10^{-3}$ is a safe target and therefore should be used in practice.

3.5 Security vulnerabilities

In AS-SPT, AST servers are vulnerable to DoS attacks. A large number of traceback queries coming from the attackers can overwhelm an AST server. One approach to defend against such DoS attacks is to strictly control the validity of the incoming query messages. That is, in AS-SPT used in a recursive querying mode, each AST server collaborates with the AST servers within R AS-hop neighbouring ASes. Therefore, an AST server can serve the traceback queries coming from these AST servers and can safely drop the others. However, determined attackers can spoof IP addresses of the neighbouring AST servers and send a large number of redundant traceback queries to an AST server. Without a defence mechanism against address spoofing-based attacks, the AST server becomes vulnerable to these type of attacks.

Fortunately, several approaches have been proposed to defend against spoofing attacks. First of all, the deployment of ingress and egress filtering may help detect such spoofing attacks (Ferguson and Senie, 2000; Killalea, 2000). Another approach may be to utilise the router level hop distance information between the neighbouring AST servers as presented by Jin et al. (2003). Here the main idea is that neighbouring AST servers could know the router level distance between each other and can use this information to decide whether a traceback query is coming from the neighbouring AST server. If the query message is coming from an attacker with a spoofed IP address, it is most likely that the router level distance between the attacker and the victim AST server will be different from the distance between the spoofed AST server and the victim AST server. Accordingly, the victim AST server may easily detect spoofing-based attack queries and drop them. Another approach is to implement a lightweight authentication

Figure 4 Extra queries per traceback request due to false positive



protocol among the neighbouring AST servers as presented by Bremler-Barr and Levy (2005). During the neighbour identification phase, AST servers can exchange pairwise secret keys to implement an authentication protocol. Later on, during the regular operation, each AST server can use the pairwise keys to authenticate themselves to each other.

Finally, we can make use of skipping in dealing with attacked AST servers. For example, suppose an attack on an AST server (say AST A) is successful at preventing a neighbouring AST server (say AST B) from accessing the attacked server AST A. In this case, the AST B can skip the server AST A and query the AST servers at the next level to continue the traceback operation.

4 Performance evaluations

We use both simulation and analytical models to verify the correctness of the proposed AS-SPT approach for tracing a single packet at the AS-level and to evaluate its performance under partial deployment scenarios. The main performance metrics we consider are: *full Success Rate* (SR), partial *detection depth* and *number of queries* per traceback request.

- We measure the full SR as the ratio of the requests for which traceback succeed to the total number of requests made by the victims whose AS has deployed traceback. If the victim's AS did not deploy, then that AS would not be able to start traceback. In other words, SR measures what is the probability of detecting the attacker's AS when the victim's AS deploys while the other ASes participate at certain deployment ratios.
- In the case of partial deployment, a traceback search may not find the complete path (full success). However, it may partially detect a path (partial success). To measure partial success, we consider *detection depth* which is defined as the number of ASes found in the traceback path. The detection depth is upper bounded by the number of AS-hops in the full attack path.
- We also measure the number of queries issued during the traceback process. If the number of traceback requests increases or the skip level increases, then reducing number of queries per traceback will have less load on the participating ASes and improve the response time. Reducing number of unnecessary queries will also avoid some false positives.

Consider an attack packet that traverses the AS path *A-B-C-D-V*, where the attacker is in AS *A* and the victim is in AS *V*. If the traceback process identifies the partial path *C-D-V*, the detection depth is 2. If the traceback process identifies the partial path *B-C-D-V*, then the detection depth is 3. For full success in which traceback identifies the attack origin AS *A*, the detection depth is 4. In practice, partial success would be valuable because providers could apply more efficient upstream filtering against ongoing DDoS attacks or could alert non-participating providers that they are involved in the transmission of the identified malicious traffic.

Results for our performance metrics depend upon the number and the location of participating ASes and upon the maximum skip level. In reporting our results, we use the 'x-level skip' term to indicate the maximum number of allowed skips. More specifically, consider an attack path as *A-B-C-D-V*. With a 1-level skip, an AST server may only skip one level. For instance if *D* does not deploy AS-SPT but *C* does, then *V* may skip over *D* and send a traceback query directly to *C*. With a 2-level skip, *V* may skip over both *C* and *D* if they do not deploy the AS-SPT method. In this case the query is sent directly to *B*.

To illustrate how the x-level skip affects the success of a traceback process, consider the case where AS *C* is not participating while all the others are. If the skip level is 0, then no skips are allowed and the traceback process ends after finding the path *C-D-V*. If the skip level is increased to 1, then queries can be sent to the predecessors of *C* and the traceback operation fully succeeds. Also note that, in the above example, because *V* does not know which of *C*'s neighbours are in the attack path, *V* must send traceback queries to *all* the neighbours of *C*. In other words, skip level along with deployment ratio will also affect the query overhead.

4.1 Simulation results

We consider three different topologies for our simulations. The first is a realistic 8998-node AS-level topology map generated from 3 weeks of CAIDA data collected from 1 to 23 June, 2004 (CAIDA, 2006). We call this topology SKITTER. We also consider a randomly generated AS-level graph with 10,000 nodes created by BRITE (2006). Since we use the Barabasi and Albert (BA) model in generating this topology, we call it BA10K. Both of the AS-level graphs obey commonly observed power-law degree characteristics of the internet. In addition, both topologies have the average AS-hop count between 3 and 4, which is consistent with observed AS-hop counts for the internet (Fei et al., 1998). Our third topology is a 30×30 mesh which we include to investigate the impact of long paths on the traceback performance.

In all topologies, we assume that each node represents an AS and that the participating ASes have the ability to answer queries with 100% accuracy (no false positives). We use two models to select participating ASes and to select the ASes of the attacker and victim:

- *Uniform model:* the uniform model does not distinguish between stub or transit ASes. We instead randomly choose 10%, 20%,..., 100% of the ASes in the topology and set them to be AS-SPT deploying ASes. We also randomly choose the ASes of the attacker and the victim in the topology map.
- *Stub-transit differentiated model:* this model differentiates between stub and transit ASes. If the degree of an AS is less than or equal to two, then it is assigned to the StubAS group; otherwise, it is assigned to the TransitAS group. According to this definition, BA10K has 4972 StubASes and 5028 TransitASes; SKITTER has 5238 StubASes and 3760 TransitASes. We then randomly select 20%, 40%, 60% and 80% of TransitASes and mark them as the participating ASes. For each TransitAS deployment rate, we randomly choose 10%, 20%,..., 100% of StubASes as the participating ASes. Finally, we randomly select the ASes of the attacker and the victim from the StubAS group.

For each topology, we run our simulation 10,000 times. Each time, we randomly select the participating ASes and the ASes of the attacker and the victim, as defined above. Note that we make sure the victim's AS is a participating one. Then, we compute the shortest path from the attacker to the victim and set that path as the attack path. We then perform the traceback process while respecting the given number of skip levels. We set the number of the skip levels to be 0, 1, 2, 3 and/or 4. For each setting, we execute the simulations and compute the success rate of the traceback processes, the average number of the queries per request, and the partial detection depth.

4.1.1 Simulation results under uniform model

Figures 5–7 give simulation results for the SKITTER, BA10K and 30 × 30 mesh topologies, respectively. Part (a) in each figure contains a histogram of the path length (AS-level hop count) distribution for that topology model. Because the SKITTER topology is constructed from the measurements of the real internet, most AS path lengths are 3 or 4 as expected. BA10K also demonstrates similar distributions. The 30 × 30 mesh topology differs substantially from the internet topology models, with most path lengths in the range of 10–30 hops and an average of 20 hops. Path length is important as it significantly affects the success rate, detection depth, and message overhead of the traceback process under different skip levels and deployment rates.

Part (b) in each figure shows the full SR. Note that full success requires the attacker's AS to deploy and/or be connected to at least one participating AS. Consequently, the rate of full success cannot exceed $2r - r^2$, where r is the deployment rate. As seen in the figures, the traceback results depend primarily upon the path length. For the internet topology models with relatively short path lengths, the success rate for 0-level skip (no skips allowed) is noticeably lower than the success rate for 1-level skip (1 skip allowed). Beyond that, the skip-level has little impact on the success rate. That is, there is little difference in the success rate between skip levels 1, 2 and 3. This is especially true for the SKITTER topology. Conversely, traceback on the 30 × 30 mesh topology with longer path lengths requires a 4-level skip or higher to achieve the same success rate as in the internet topologies.

Part (c) of each figure shows the number of messages per traceback request under different skip levels and deployment ratios. When the skip level is 0, only 1 query is sent per hop and the maximum number of queries equals to the path length. However, when a non-participating AS is skipped over, queries are sent to its N neighbours. Skipping over a second adjacent AS with M neighbours requires that

Figure 5 Simulation results using SKITTER network topology

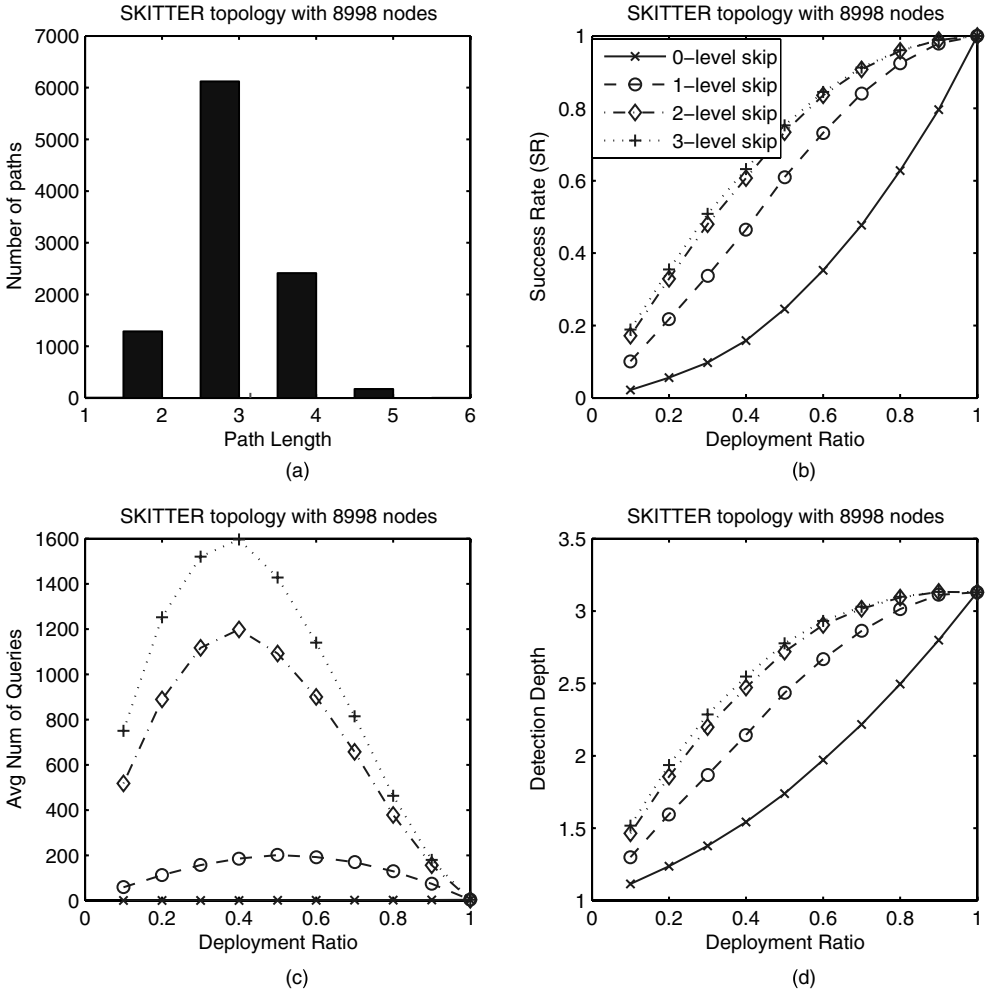
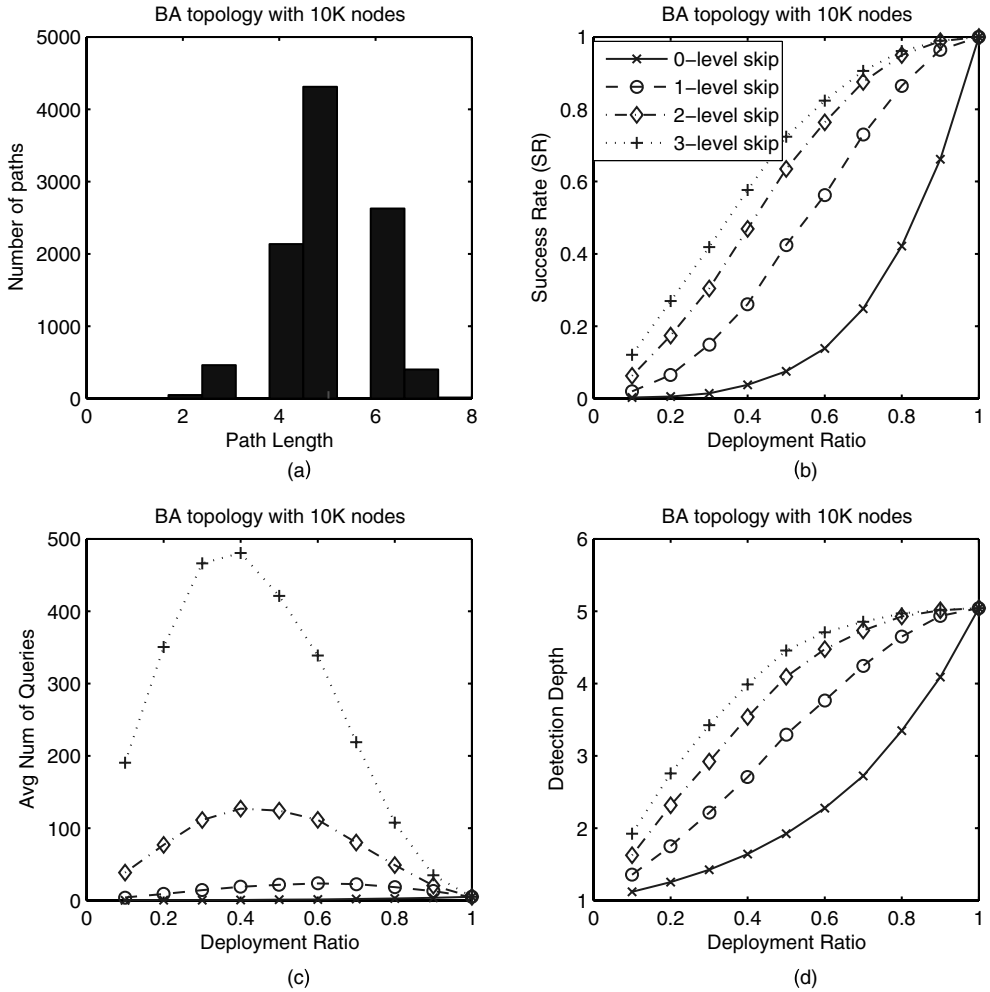


Figure 6 Simulation results using BA network topology



the victim send $N \times M$ queries. As a result, message overhead increases dramatically with the skip level for most deployment ratios. At very low deployment ratios, there are few messages, because the victim AS has a high probability of not participating. The message overhead is highest for the mid-range of deployment ratios, because in many cases the victim AS will generate a traceback request that will need to skip over several adjacent ASes in the path. At high deployment ratios, most ASes in the path are participating, so the number of messages decreases.

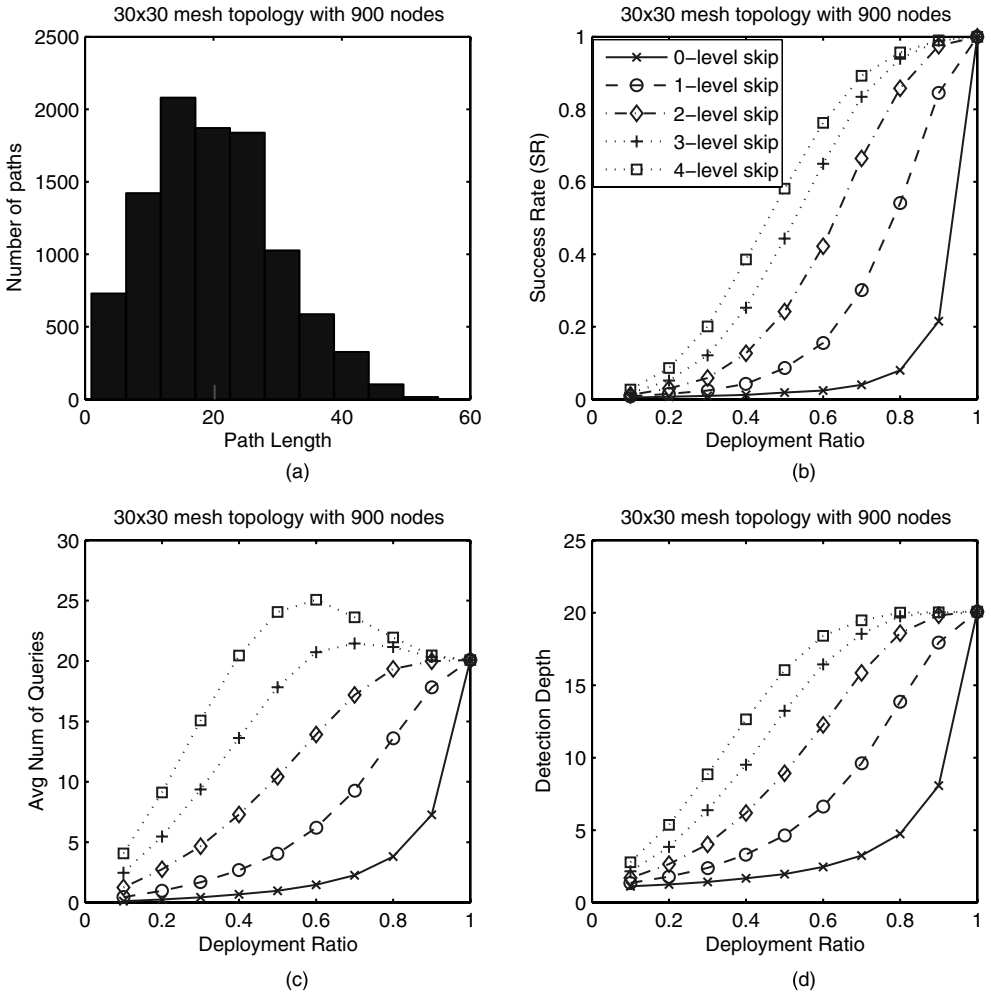
Finally, Part (d) of each figure illustrates the extent to which the traceback process finds a partial path to the attacker. As with full success, the skip level has a small effect on partial success for internet-like topologies with smaller path lengths. Conversely, the 30×30 mesh topology shows poor traceback results unless the skip level is relatively high.

4.1.2 Simulation results under stub-transit differentiated model

Figures 8 and 9, respectively, give simulation results for the SKITTER and BA10K topologies under different deployment ratios for Transit and Stub ASes. Both topologies show the same general results. Two observations stand out. Firstly, participation by transit ASes is more critical than participation by stub ASes. In general, the success rate starts near the transit deployment rate and increases linearly with stub deployment rate. Consider the SKITTER topology in Figure 8. For a transit AS deployment rate of 0.80 and a skip level greater than 0, the success rate ranges from 0.80 to over 0.95. For a transit AS deployment rate of 0.60, the success rate ranges from near 0.60 to over 0.75 at skip level 1 and from 0.60 to near 1.0 at higher skip levels. We explain this as follows. Traceback proceeds through transit ASes and will succeed if the final transit AS deploys, provided the skip level is sufficiently high. Consequently, the success rate starts at or above the transit deployment rate for skip level greater than 0. The other feature of success rate is its linear dependence upon stub deployment rate. Consider the case where the final transit AS does not deploy. Traceback can still succeed if the final transit AS is skipped over and the query is sent to the origin AS (a stub AS). In these cases, the success rate depends linearly upon stub deployment rate.

The second key observation is the impact of the skip level on success rate and message overhead. For both topologies, a skip level of 1 incurs little overhead cost regardless of deployment rate. Overhead dramatically increases at skip levels of 2 and higher. For success rate, the dramatic improvement occurs between skip levels 0 and 1. Combining these two observations, we find that setting the skip level to 1 provides a good balance between success rate and overhead.

Figure 7 Simulation results using 30x30 mesh network topology



4.2 Analytical results

In order to supplement the simulation results, we conduct an analytical study on the aforementioned performance metrics. Due to the difficulties in building a mathematical model to study a generic network topology, we use a simple path as the network topology between the attacker AS and the victim AS and call it line topology. The topology we consider can be represented as: VictimAS-AS1-AS2-AS3-$\cdots$-ASn-AttackerAS. We assume that VictimAS deploys AS-SPT; otherwise, we could not start the traceback process. We also assume that other ASes independently deploy AS-SPT with a probability r.

When no skip is allowed, VictimAS sends a query to AS1 with probability r. It will then send next query to AS2 with probability r^2 and so on. If 1-level skip is allowed, then VictimAS will send a query to AS2 over two different paths:

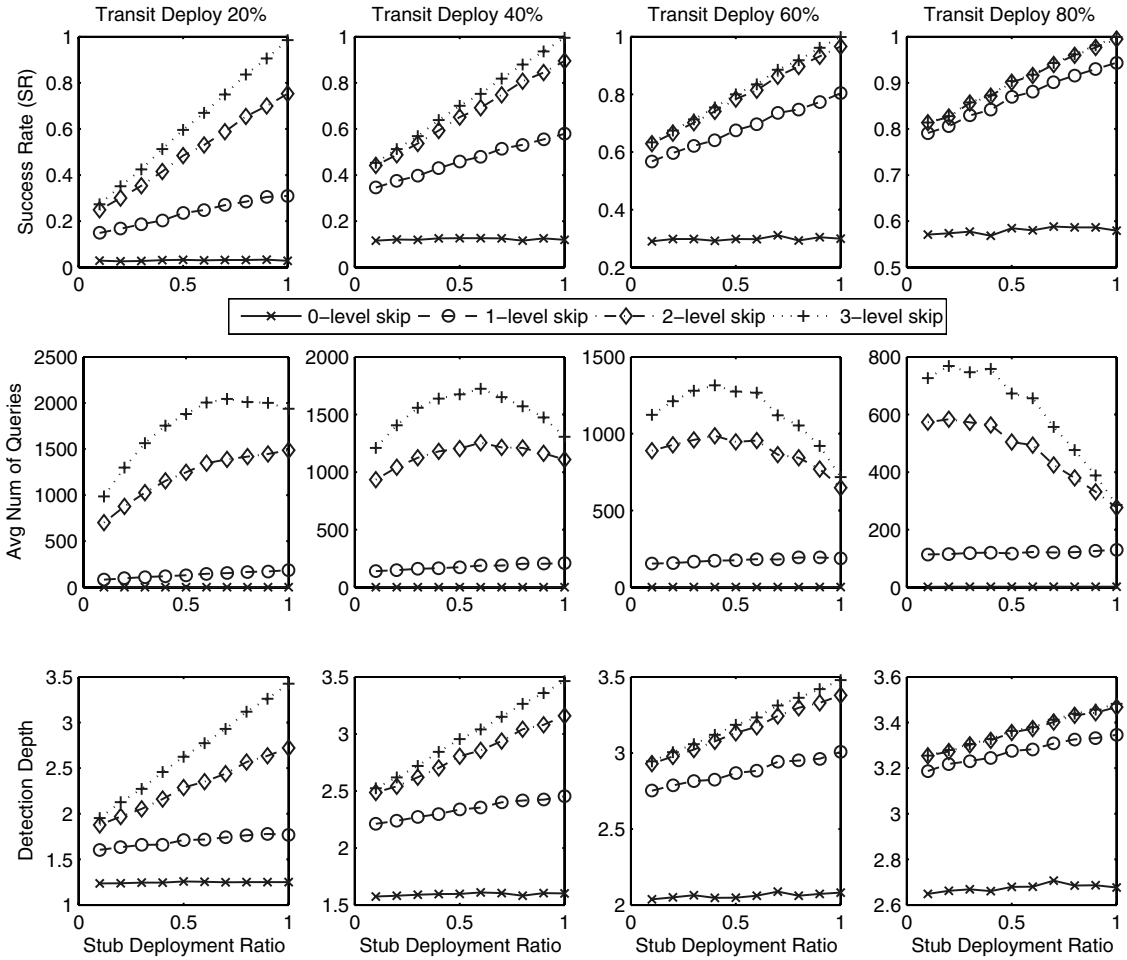
- $\langle$ VictimAS, AS1 and AS2 $\rangle$: both AS1 and AS2 have deployed AS-SPT, thus the probability of following this path is r^2.
- $\langle$ VictimAS, (skip AS1), AS 2$\rangle$: AS2 deploys but AS1 does not (and hence can be skipped), so the probability for this way is qr, where $q = 1 - r$.

As a result, the total probability to send a query from VictimAS to AS2 is $r^2 + qr$. Likewise, VictimAS will send one query to AS3 with probability $r^3 + qrr + rqr$ and so on. The corresponding recurrence relationship can be visualised using the following table:

0	*1*	*2*	*3*	*4*	...	*n−1*	*n*
1	r	r^2	r^3	r^4	...	r^{n-1}	r^n
		qr	qrr	qrr^2	...	qrr^{n-3}	qrr^{n-2}
			rqr	$rqrr$	...	$rqrr^{n-4}$	$rqrr^{n-3}$
				qrr^2	...	qrr^{n-3}	qrr^{n-2}
				$qrqr$	...	$qrqrr^{n-5}$	$qrqrr^{n-4}$
					⋮	⋮	⋮
a_0	a_1	a_2	a_3	a_4	...	a_{n-1}	a_n

Let the sum of terms in column k be a_k, $k = 0, 1, 2, \ldots, n$. In essence, a_k is the probability that a traceback process will reach node k. Therefore, we will make use of it when computing the three performance metrics described in the beginning of this section. Before that, we first explain how to compute a_k for one or more skip levels.

Figure 8 Simulation results using different deployment ratios at the core and stub ASes in SKITTER



4.2.1 Computing a_k

After carefully analysing the above table given for 1-level skip, we can compute a_k by simply using the following recurrence relation:

$$a_k = qra_{k-2} + ra_{k-1} \tag{1}$$

with $a_0 = 1$, and $a_1 = r$. Using the well-known techniques from combinatorics (Tucker, 1995), we can solve this recurrence relation as follows. Setting $a_k = \alpha^k$, we obtain the characteristic equation as:

$$\alpha^k = qr\alpha^{k-2} + r\alpha^{k-1}$$

Dividing each side by α^{k-2}, we get

$$\alpha^2 = qr + r\alpha$$

This may be written as

$$\alpha^2 - r\alpha - qr = 0$$

Solving this quadratic equation, we get two roots:

$$\alpha_1 = \frac{r - \sqrt{r^2 + 4qr}}{2} \quad \text{and} \quad \alpha_2 = \frac{r + \sqrt{r^2 + 4qr}}{2}$$

As a result, the general solution will be

$$a_k = A_1\alpha_1^k + A_2\alpha_2^k$$

Using the initial conditions $a_0 = 1$ and $a_1 = r$, we get

$$a_0 = A_1\alpha_1^0 + A_2\alpha_2^0 \Rightarrow A_1 + A_2 = 1$$

$$a_1 = A_1\alpha_1^1 + A_2\alpha_2^1 \Rightarrow \alpha_1 A_1 + \alpha_2 A_2 = r$$

After solving these equations for A_1 and A_2, we obtain:

$$a_k = \frac{r - \alpha_2}{\alpha_1 - \alpha_2}\alpha_1^k + \frac{r - \alpha_1}{\alpha_2 - \alpha_1}\alpha_2^k \tag{2}$$

The above discussion is valid in case of 1-level skip. Fortunately, we can generalise the recurrence relationship (1) for other skip levels as well. For example, when 2-level skip is allowed, we will have

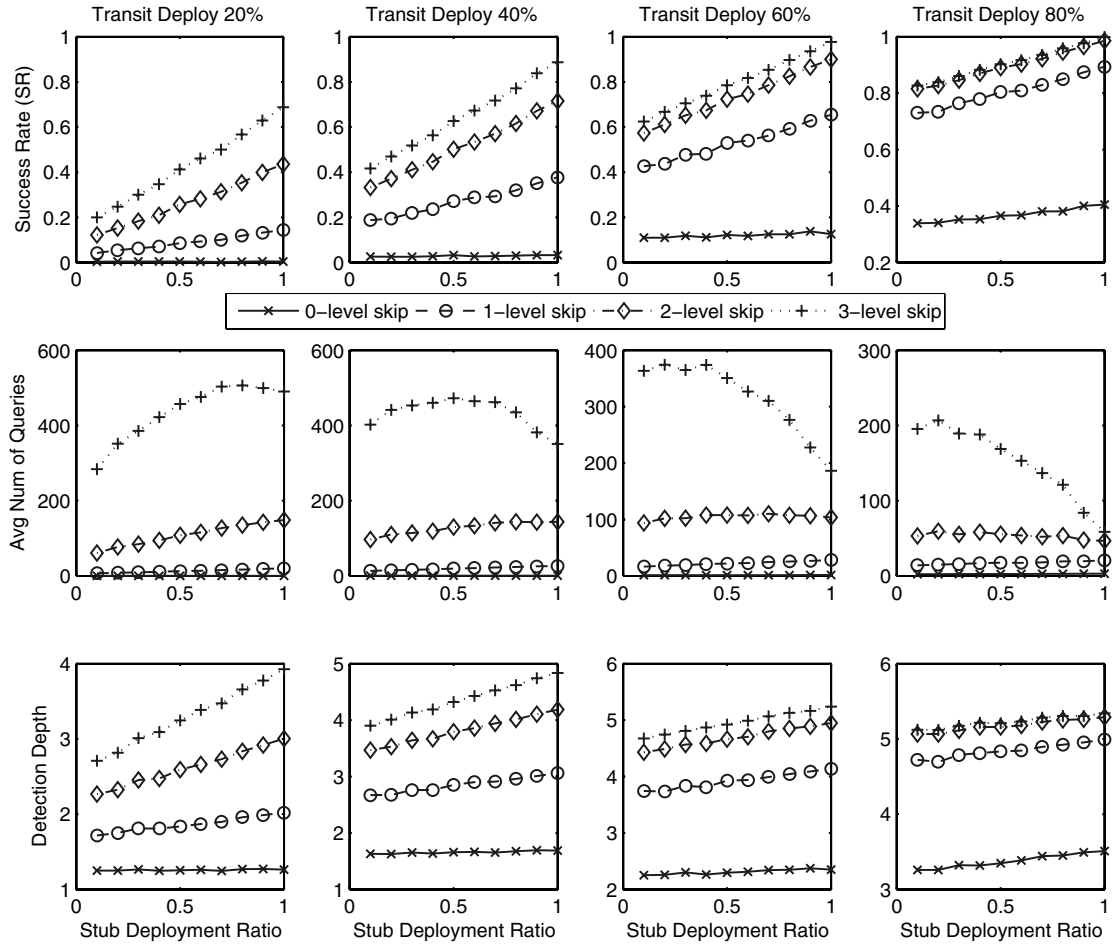
$$a_k = qqra_{k-3} + qra_{k-2} + ra_{k-1}$$

with $a_0 = 1$, $a_1 = r$ and $a_2 = qra_0 + ra_1$. Similarly, when 3-level skip is allowed, we will have

$$a_k = qqqra_{k-4} + qqra_{k-3} + qra_{k-2} + ra_{k-1}$$

with $a_0 = 1$, $a_1 = r$, $a_2 = qra_0 + ra_1$ and $a_3 = qqra_0 + qra_1 + ra_2$ and so on. However, in the case of

Figure 9 Simulation results using different deployment ratios at the core and stub ASes in BA10K



more than 1-level skip, obtaining the closed form solutions similar to (2) will be difficult and complex if not impossible.

4.2.2 *Computing performance metrics*

As mentioned before, a_k is the probability that the traceback process comes up to node k. To be fully successful in the line topology, the traceback request must come to node n and/or node $n-1$. The probabilities for these two cases are respectively given by a_n and a_{n-1}. Thus, we can conclude that the success rate of the traceback process in line topology is equal to

$$\text{SR} = a_n + a_{n-1} - a_n a_{n-1}$$

For the average number of queries, we can again make use of a_k. Clearly, VictimAS will send one query to each ASk with a probability a_k, $k = 1, 2, \ldots, n$. Thus, we can compute the expected number of queries using

$$E[\text{numberofqueries}] = \sum_{k=1}^{n} a_k$$

In case of 1-level skip, using the closed form equation (2), we can obtain the following closed form formula for

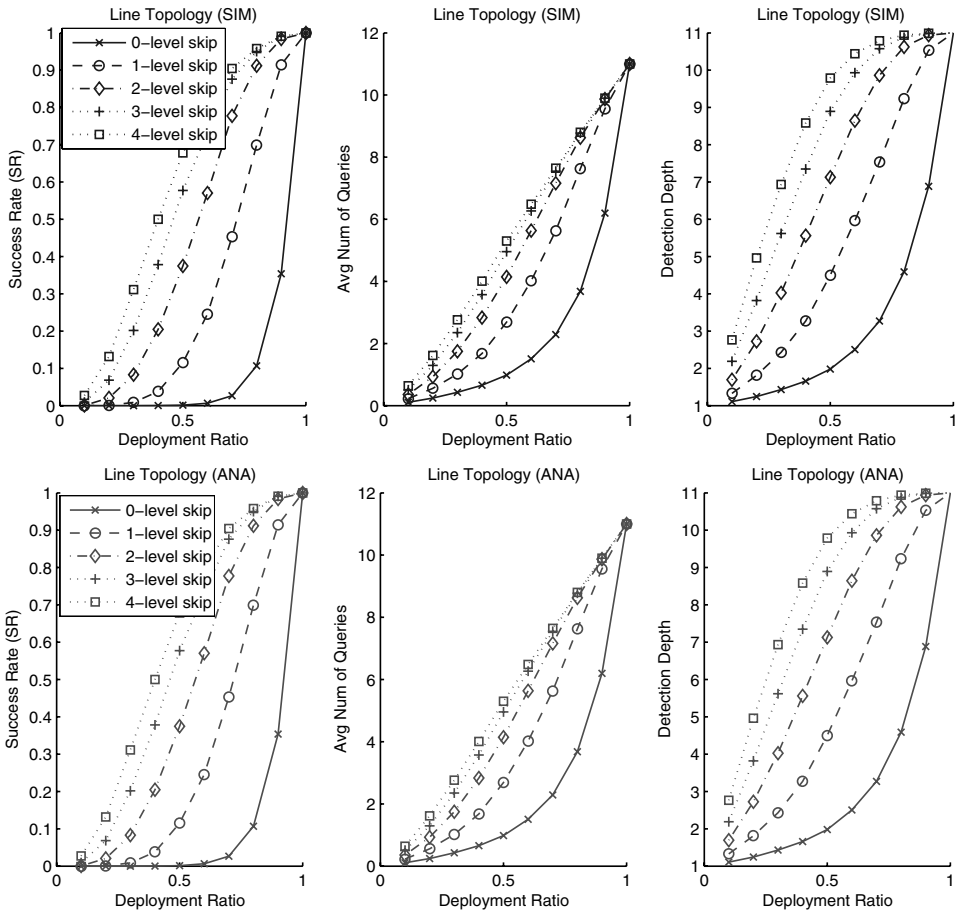
$$E[\text{numberofqueries}] = \frac{r-\alpha_2}{\alpha_1-\alpha_2}\frac{1-\alpha_1^{n+1}}{1-\alpha_1} + \frac{r-\alpha_1}{\alpha_2-\alpha_1}\frac{1-\alpha_2^{n+1}}{1-\alpha_2} - 2$$

For other skip levels, we simply use numerical computation. Finally, we can compute the average of the partial detection depth as follows. When the traceback comes to node k with probability of a_k, we will know that the attack packet comes from $k+1^{st}$ AS. In other words, we detect the path up to $k+1$ with probability a_k. This will be true until $k = n-1$, when $k = n$ the detection depth will be n. As a result, we can compute the average of detection depth using the following formula:

$$\text{Detection depth} = na_n + \sum_{k=0}^{n-1}(k+1)a_k$$

After computing a_k using the closed form formulas or a simple procedure implementing the recurrence relation, we can easily compute the performance metrics as explained above. In fact, we computed these metrics for $n = 11$ and present them in the bottom row of Figure 10. We also simulated the line topology with $n = 11$ and obtained the same results, as shown in the top row of Figure 10. Analytical model also allowed us to verify and validate our simulation experiments.

Figure 10 Simulation and analytical results using line topology with $n = 11$



5 Conclusion and future work

In this paper, we have proposed an AS-SPT mechanism and investigated various practical issues including partial deployment, packet transformation, false positives and security vulnerabilities. Using both simulations and analytical methods, we extensively investigated the effectiveness of the proposed mechanism under various partial deployment scenarios. We have explored the relationship between the deployment ratio in the network and the full or partial success rate of finding attackers. We have also characterised the trade-offs between the success rate in tracing individual packets and protocol overheads.

In our simulations, we have considered various topologies along with both uniform and stub-transit differentiated deployment scenarios. Our simulation results show that in internet-like topologies with short path lengths, AS-SPT can successfully determine full or partial AS paths with relatively little communication overhead. Conversely, a mesh topology with longer path lengths has lower success rates and higher communication overheads. These results suggest that a primary benefit of AS-levels traceback over router-level traceback is due to the shorter AS path lengths. In our analytical model, we have considered an attack path as a line topology and computed success rate, detection depth and query overhead. This model provides a lower bound on the performance metrics. In addition, it allowed us to verify and validate our simulation programme.

For future work, we plan to develop formal protocol specifications for utilising the communication between participating AST servers and present our ideas to standardization bodies like IETF. We will also consider developing new online and/or off-line mechanisms to efficiently and accurately discover the neighbourhood information for the participating ASes. To further reduce the query overhead, we will also investigate mechanisms such as hop-count filtering and AS-level filtering.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. We would also like to thank Trinh Le from The University of Texas at San Antonio and Nathan A. Price from Southwest Research Institute for helping to develop some of the simulation programmes. Sandra G. Dykes thanks SwRI for supporting her work in part by internal research and development grant 10.R9446.

References

Belenky, A. and Ansari, N. (2003) 'On IP traceback', *IEEE Communications Magazine*, Vol. 41, No. 7, pp.142–153.

Bloom, B. (1970) 'Space/time trade-offs in hash coding with allowable errors', *Communications of ACM*, Vol. 13, No. 7, pp.422–426.

Bremler-Barr, A. and Levy, H. (2005) 'Spoofing prevention method', *Proceedings of IEEE Infocom*, Miami, FL.

BRITE (2006) 'BRITE: Boston university representative internet topology generator', Available at: www.cs.bu.edu/brite.

CAIDA (2006) 'Cooperative association for internet data analysis', Available at: http://www.caida.org/.

Fan, L., Cao, P., Almeida, J., and Broder, A. (2000) 'Summary cache: a scalable wide-area web cache sharing protocol', *IEEE/ACM Transactions on Networking*, Vol. 8, No. 3, pp.281–293.

Fei, A., Pei, G., Liu, R. and Zhang, L. (1998) 'Measurements on delay and hop-count of the internet', *Proceedings of IEEE GLOBECOM*, Sydney, Australia.

Ferguson, P. and Senie, D. (2000) *Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing*, RFC 2827.

Gao, Z. and Ansari, N. (2005) 'Tracing cyber attacks from the practical perspectives', *IEEE Communications Magazine*, Vol. 41, No. 5, pp.123–131.

Garber, L. (2000) 'Denial-of-service attacks rip the internet', *IEEE Computer*, Vol. 33, No. 4, pp.12–17.

Gong, C. and Sarac, K. (2005) 'IP traceback based on packet marking and logging', *Proceedings of IEEE International Conference on Communications (ICC)*, Seoul, Korea.

Gorg, C., Lamers, E., Fus, O. and Heegaard, P.E. (2001) 'Rare event simulation', *COST report 257*, Available at: http://www.item.ntnu.no/∼poulh/publications/res-cost.pdf.

Jin, C., Wang, H. and Shin, K.G. (2003) 'Hop-count filtering: an effective defense against spoofed ddos traffic', *Proceedings of the Tenth ACM Conference on Computer and Communications Security*, New York, NY, pp.30–41.

Killalea, T. (2000) *Recommended internet Service Provider Security Services and Procedures*, RFC 3013.

Lee, T., Wu, W. and Huang, W. (2004) 'Scalable packet digesting schemes for IP traceback', *Proceedings of IEEE International Conference on Communications (ICC)*, Paris, France.

Li, J., Sung, M., Xu, J., Li, L. and Zhao, Q. (2004) 'Large-scale IP traceback in high-speed internet: practical techniques and theoretical foundation', *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA.

McCreary, S. and Claffy, K. (2000) 'Trends in wide area IP traffic patterns: a view from Ames internet exchange', *ITC Specialist Seminar on IP Traffic Modeling, Measurement and Management*.

Moore, D., Voelker, G. and Savage, S. (2001) 'Inferring internet denial of service activity', *Proceedings of USENIX Security Symposium*, Washington, DC.

Moriarty, K.M. (2005) *Incident Handling: Real-Time Inter-Network Defense*, Internet draft, Available at: http://www.ietf.org/internet-drafts/draft-ietf-inch-rid-05.txt.

Sager, G. (1998) 'Security fun with OCxmon and cflowd', *Presentation at the Internet 2 Working Group Meeting*, San Diego, CA, USA Available at: http://www.caida.org/funding/ngi/content/security/1198/.

Sanchez, L., Milliken, W., Snoeren, A., Tchakountio, F., Jones, C., Kent, S., Partrige, C., and Strayer, W. (2001) 'Hardware support for a hash-based IP traceback', *Proceedings of the Second DARPA Information Survivability Conference and Exposition*, Anaheim, CA, pp.146–152.

Snoeren, A., Partridge, C., Sanchez, L., Jones, C., Tchakountio, F., Schwartz, B., Kent, S. and Strayer, W. (2002) 'Single-packet IP traceback', *IEEE/ACM Transactions on Networking*, Vol. 10, No. 6, pp.721–734.

Strayer, W., Jones, C., Tchakountio, F., Snoeren, A., Schwartz, B., Clements, R., Condell, M. and Partridge, C. (2003) 'Traceback of single IP packets using SPIE', *Proceedings of the DARPA Information Survivability Conference and Exposition*, Washington, DC.

Tucker, A. (1995) *Applied Combinatorics*, John Wiley and Sons, Inc.

Notes

[1]An abridged version of this paper was presented at the Symposium on Computer and Network Security in IEEE GLOBECOM 2005, Nov. 28 – Dec. 2 2005, St. Louis, Missouri, USA.

[2]A recent study by [McCreary and Claffy, 2000] on wide-area traffic patterns shows that less than 3% of IP traffic undergoes various transformations that can affect the fields we consider.