Towards Stateless Single-Packet IP Traceback

Rafael P. Laufer[†], Pedro B. Velloso[‡], Daniel de O. Cunha[‡], Igor M. Moraes[§],
Marco D. D. Bicudo[§], Marcelo D. D. Moreira[§], and Otto Carlos M. B. Duarte[§]

[†]University of California at Los Angeles
Los Angeles, USA

[‡]Université Pierre et Marie Curie - Paris VI
Paris, France

[§]Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil

***Abstract*— The current Internet architecture allows malicious nodes to disguise their origin during denial-of-service attacks with IP spoofing. A well-known solution to identify these nodes is IP traceback. In this paper, we introduce and analyze a light-weight single-packet IP traceback system that does not store any data in the network core. The proposed system relies on a novel data structure called Generalized Bloom Filter, which is tamper resistant. In addition, an efficient improved path reconstruction procedure is introduced and evaluated. Analytical and simulation results are presented to show the effectiveness of the proposed scheme. The simulations are performed in an Internet-based scenario and the results show that the proposed system locates the real attack path with high accuracy.**

I. Introduction

Denial-of-service (DoS) attacks are currently the fifth major cause of financial loss due to cybercrime [1]. Originally, these attacks were launched using one or a small number of hosts to flood the victim with spoofed service requests. The goal was to deplete its resources with bogus traffic, leaving no bandwidth for legitimate traffic. These attacks, however, require that the attacking hosts send a huge volume of traffic towards the victim, which could be easily traced using simple traffic analysis techniques [2], [3]. Currently, distributed DoS (DDoS) attacks composed of 1.5 million computers are already a reality [4]. If a large enough number of hosts is used to disable a common victim, each attacking host may generate only a tiny amount of traffic. The aggregate traffic is then responsible for wasting the victim's resources and disabling its services. This strategy makes it much harder to trace the true generators of traffic and still keeps the anonymity of the attackers.

Another particular attack that also challenges currently available defense techniques is the *single-packet* DoS attack [5]–[9]. Instead of flooding the victim with lots of service requests, such attacks are based on specific vulnerabilities triggered by a carefully crafted packet. As a result, they are much easier to be conducted than flooding-based attacks, since only one packet must be generated to deny the victim's service. Therefore, ideal defense techniques must be effective against both small-flow and single-packet attacks.

This work has been supported by CNPq, CAPES, FAPERJ, FINEP, FUNTTEL, RNP, and UOL.

One defense approach is to inhibit DoS attacks from even happening. These incidents only happen because it is possible for attackers to hurt the victims and still remain anonymous. For instance, distributed DoS attacks are usually conducted using multiple "zombie" machines remotely controlled by the attacker. Zombies rely on spoofed source addresses to properly disguise their true origin [10]. Spoofed source addresses add an extra layer of protection for the attacker without any additional cost. If, however, the network is capable of tracing each packet back to its true source, zombies can be easily identified. Other techniques, such as stepping-stone detection [11], must then be applied to actually identify the computer used to launch the attack. Once the network can trace each packet back to its true originator, attacks can no longer be conducted anonymously and legal actions can finally be taken.

We address in this paper the identification of computers that directly generate attack traffic, regarded as the IP traceback problem [12]. Tracing an attack back to its source is essential in any kind of attack that uses spoofing techniques. Single-packet attacks, however, are very hard to trace and most of the proposed traceback schemes assume that attacks are composed of large flows [2], [3], [12]–[15]. The basic idea of such schemes is to encode path information in the attack packets themselves to allow the victim to identify the generators of traffic. They take advantage of large flows to distribute the path information among the different packets of the flow and therefore reduce the per-packet overhead. These schemes, however, perform poorly in small-scale distributed attacks [16] and can not trace single-packet DoS attacks, since the path information is divided among different packets.

In order to trace a single packet, we must choose from two basic options. We can either have each packet storing the complete information about the path taken or have each router storing information about every forwarded packet [3], [17]. The tradeoff here is to have either additional overhead in the packet header or to keep per-packet state at routers.

The main challenge is to design a single-packet IP traceback scheme that is suitable for *high-speed* networks. In order to fulfill the requirements of these networks, practical traceback schemes must meet two basic criteria. First, little processing overhead should be added to routers. Increasing the per-packet processing time directly affects the router throughput and, therefore, it should be kept as low as possible. Secondly, no

information must be stored in the network core [18]. As the speed of the network increases, the volume of auditing data may become too large to be stored even for a small time period. As a consequence, network routers should not keep any per-packet state. Our proposal deals with both problems in a clever way.

In this paper, we present a light-weight and stateless approach to the IP traceback problem. Our proposal has the advantage of tracing an attack by extracting the path information from a *single packet* without *any state* in the network core. In our proposal, the additional processing overhead for routers is composed of only two bitwise logical operations. The proposal consists of using a Bloom Filter [19] integrated into the packet header to store the IP addresses of traversed routers in a compact form. Later, the victim initiates a path reconstruction procedure to identify the actual attack source. In order to prevent the attacker from interfering with the tracing, we propose a generalization of Bloom Filters and use it in the packet header to store the traversed routers. The key idea of the so-called Generalized Bloom Filters (GBF) is to use hash functions that also reset bits during element insertions. The tradeoff cost is that false negatives are introduced with the proposed generalization. A false negative in the path reconstruction procedure means not detecting a router actually traversed by the packet.

In a companion paper [20], we sketched the initial idea of our design and primary results were derived. In this paper, we extend our previous work by providing an improved path reconstruction procedure that completely eliminates the false negatives introduced by the Generalized Bloom Filter. Additionally, we also show that we locate the attacker with very high accuracy during simulations in an Internet-based topology and that no false negatives happen with the proposed reconstruction procedure.

This paper is organized as follows. The proposed IP traceback system is introduced in Section II. The analysis of our generalization of Bloom Filters is briefly explained in Section III. The improved reconstruction procedure is introduced in Section IV. We present simulation results of the new reconstruction procedure in an Internet-based topology in Section V. Section VI details the related work and conclusions are finally presented in Section VII.

II. Node Digesting

In this section, we present our stateless single-packet IP traceback system. The proposal is based on a packet-marking technique to avoid storing state at routers. In summary, each node inserts a mark into routed packets to notify the victim of its presence on the path. Upon receiving an attack packet, the victim uses the node-made markings to reconstruct the entire path. Route auditing is performed by inserting only the *digests* of IP addresses into the packets rather than IP addresses themselves. A built-in Bloom Filter [19] is used to reduce the required per-packet space and keep the size of the information inserted into the packet constant. The size limitation is important to avoid both the appending processing overhead and packet fragmentation. We also introduce the so-called Generalized Bloom Filter to prevent the attacker from forging node digests and causing backtracing failures.

The packet-marking procedure for this scheme is quite simple. Just before forwarding a packet, the router inserts the IP address of its output interface into the Bloom Filter of the packet. One important advantage of this marking procedure is its low additional processing overhead. In fact, no hash calculations need to be made on a per-packet basis. The hashes of the IP addresses of every router interface may be calculated in advance and stored in a series of so-called "mask" registers. These registers can be seen as Bloom Filters with only one element inserted: the IP address of the interface. When a packet is about to be forwarded, the filter of the packet is simply updated by taking the result of a bitwise OR of itself and the output-interface "mask" register.

To reconstruct the attack path, the following procedure is performed. First, the victim tests the membership of all neighbor routers in the Bloom Filter of the received attack packet. The one recognized by the filter is identified as the upstream router and integrated into the attack path. Then, this upstream router receives the respective Bloom Filter from the victim to continue the reconstruction procedure. It then checks which neighbor router is also recognized by the filter, identifying the next upstream router. This procedure is recursively repeated on each router to reconstruct the actual path traversed by the packet. When no neighbor router is recognized, the procedure stops and the router performing the tests is considered the source of the attack. Fig. 1 depicts a path reconstruction procedure starting at a victim V towards an attacker A. First, the attacker sends a packet to the victim that traverses the path (R_5, R_4, R_2, R_1). Upon receiving the attack packet, the victim initiates the reconstruction procedure by testing R_1 against the filter of the received packet (1). Since R_1 passes the test, it receives the filter from V and continues the reconstruction procedure. Accordingly, R_1 tests the membership of R_2 and R_3 in the received filter (2). Since only R_2 is recognized, the filter is sent to R_2 which performs the same test with its neighbor R_4 (3). The router R_4 is also recognized and it checks the membership of R_3 and R_5 (4), but only R_5 is a legitimate element. Finally, R_5 tests the membership of R_7 in the filter (5). A negative response is returned and the reconstruction procedure is complete. A message is then sent from R_5 back to V.

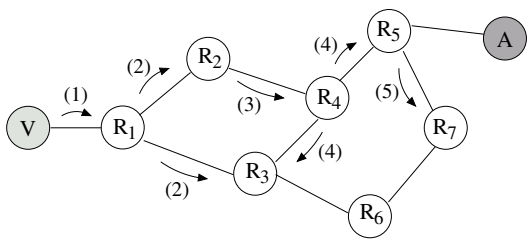

Fig. 1. Example of the path reconstruction procedure.

Remarkable advantages come from the adoption of this approach. First, the complete route of each packet can be *individually determined*. This is the goal of an ideal IP traceback system since it allows the system to be as scalable as

possible and to identify each source of a distributed or single-packet attack. Additionally, the proposed stateless approach guarantees that *no information* at all is stored in the network infrastructure. All traceback data is stored at the victim, who chooses to hold it or not according to the local security policy. Moreover, the proposed system not only avoids the appending processing overhead and packet fragmentation, but also introduces very low additional overhead to the forwarding procedure. In fact, only a per-packet bitwise OR operation is needed. Another advantage is the ability of tracing an attack long after it is over and without any help from network operators. The whole reconstruction procedure can be fully automated and independent of manual intervention.

On the other hand, this approach suffers the same drawbacks of other approaches [12]–[17]. First, as with any traceback system, routers are required to cooperate in packet marking. If few routers do not mark the packets, gaps in the reconstructed route are likely to occur and the attack source might not be found. Techniques such as expanding-ring search, however, could be employed in the reconstruction procedure to overcome this issue at the cost of a few additional false positives. Further, the attacker himself is not identified by the system; in fact, only the router closest to the attacker is exposed. After identifying the attacking router, further efforts are required to reveal the interface from which the attack traffic comes. Finally, the adoption of a Bloom Filter introduces false positives into the attack path. During the reconstruction procedure, a false positive implies the incorrect integration of a router into the attack path. If this probability is small enough, the occurrence of false positives does not significantly impact on the reconstruction. There would be some concurrent routes for the same packet but the number of possible attackers would still be small. Nevertheless, since the attacker controls the initial content of the packet, he can fill all the filter bits with 1. By saturating the filter, every router is integrated into the attack path during the reconstruction procedure, making impossible to distinguish the real path. We define this technique as an *all-one attack*.

Therefore, in order to minimize misleading techniques and to make the system less dependent on the initial content of the filter, we propose a generalization of the Bloom Filter. The basic idea of the so-called Generalized Bloom Filter (GBF) is to employ both hash functions that set and hash functions that reset bits. In the GBF, the false-positive probability is upper bounded and it does not depend on the initial condition of the filter. On the other hand, false negatives, which do not exist in standard Bloom Filters, are now introduced with this generalization. We note however that the false-negative probability of the GBF is also upper bounded and do not depend on the initial content of the filter. In Section III, the GBF is briefly described and its main results are summarized.

III. The Generalized Bloom Filter

The Generalized Bloom Filter (GBF) is a data structure used to represent a set $S = \{s_1, s_2, \ldots, s_n\}$ of n elements in a compact form. It is constituted by an array of m bits and by $k_0 + k_1$ independent hash functions $g_1, g_2, \ldots, g_{k_0}, h_1, h_2, \ldots, h_{k_1}$ whose outputs are uniformly distributed over the range $\{0, 1, \ldots, m-1\}$. The GBF is built in a similar way to the standard filter. Nevertheless, the initial value of the bits of the array is not restricted to 0 anymore. In the GBF, these bits can begin with any value. The proportion of bits initialized to zero in the GBF is defined as p_0. For each element $s_i \in S$, the bits corresponding to the positions $g_1(s_i), g_2(s_i), \ldots, g_{k_0}(s_i)$ are reset and the bits corresponding to the positions $h_1(s_i), h_2(s_i), \ldots, h_{k_1}(s_i)$ are set. In the case of a collision between a function g_i and a function h_j within the same element, we arbitrate that the bit is always reset. The same bit can be set or reset several times without restrictions. After inserting the elements, membership queries can be easily made. To check if an element x is in S, we check if the bits of the array corresponding to the positions $g_1(x), g_2(x), \ldots, g_{k_0}(x)$ are all reset and if the bits $h_1(x), h_2(x), \ldots, h_{k_1}(x)$ are all set. If at least one bit is inverted, then $x \notin S$ with high probability. In the GBF, it is possible that an element $x \in S$ may not be recognized as an element of the set, creating a false negative. Such anomaly happens when at least one of the bits $g_1(x), g_2(x), \ldots, g_{k_0}(x)$ is set or one of the bits $h_1(x), h_2(x), \ldots, h_{k_1}(x)$ is reset by another element inserted afterwards. On the other hand, if no bit is inverted, then $x \in S$ also with high probability. In fact, an element $x \notin S$ may be recognized as an element of the set, creating a false positive. A false positive occurs when the bits $g_1(x), g_2(x), \ldots, g_{k_0}(x)$ are all reset and the bits $h_1(x), h_2(x), \ldots, h_{k_1}(x)$ are all set due to other actually inserted elements or to the initial condition of the array.

A. Application

In the proposed IP traceback system, the elements inserted into the GBF of the packet are in fact the IP addresses of traversed routers. Therefore, the number of elements n represents the number of routers traversed by the attack packet. Moreover, the size of the bit array m is precisely the number of bits preallocated in the packet header for the GBF. Further, k_0 and k_1 are the number of hash functions that reset and set bits on each hop, respectively. These hash functions must be the same on every router to allow path reconstruction later. At last, p_0 represents the initial fraction of bits reset in the GBF. This parameter is in control of the attacker, who is responsible for creating the attack packet and setting the initial content of the filter.

The previous router implementation slightly changes with the adoption of a GBF. Bits now are set and reset and therefore a single bitwise OR operation is not enough to update the filter. Instead, we need a bitwise OR operation followed by a bitwise AND operation to set and reset the indicated bits, respectively. Accordingly, two "mask" registers are required per router interface. To configure the "mask" registers, the interface IP address is used as the input of the hash functions. The first register is filled with zeros and the bits indicated by the functions h_j are set; the second register is filled with ones and the bits indicated by the functions g_i are reset. When the packet is about to be forwarded, its GBF field is initially updated with the result of a bitwise OR of itself and the first

output-interface register. A bitwise AND operation of itself and the second output-interface register follows to complete the update.

The order of the operations is a result of the adopted priority. In fact, if one decides to give precedence to the functions h_j over the functions g_i, the bitwise AND operation must be done before the bitwise OR operation.

B. Upper-bounded False Positives and False Negatives

A complete discussion of the Generalized Bloom Filter and the analytical expressions of the false-positive and false-negative probabilities is presented in [21]. For now, it is sufficient to say that the both the probability of false positives and the probability of false negatives of the GBF are upper-bounded. These bounds depend solely on the number of hash functions used, k_0 and k_1, and on the relative size of the filter m/n. Since the only parameter the attacker can interfere with is the initial condition of the GBF p_0 (and not the other predefined parameters k_0, k_1, and m/n), the action of an attacker in creating false positives and false negatives is limited. This property is important to avoid the all-one attack to which standard Bloom Filters are susceptible (see Section II).

In constrast, the standard Bloom Filter does not provide any bound on the false-positive probability and simply initializing it to all ones disrupts the reconstruction procedure. An advantage of standard Bloom Filters, however, is that it does not yield false negatives.

IV. Improved Reconstruction Procedure

While using a GBF limits the action of the attacker in generating false positives, it also introduces false negatives (even though limited) in the path reconstruction procedure. A false negative means not detecting a router actually traversed by the attack packet. For instance, let router R_4 be a false negative during the reconstruction procedure depicted in Fig. 1. That is, R_4 is not recognized as an element of the filter during the membership test made by R_2 (2). In that case, routers R_3 and R_5 are not even checked (3) since R_4 was not integrated into the attack path. As a consequence, router R_5 is not integrated into the attack path either. Therefore, just one false negative is enough to stop the reconstruction procedure and avoid finding the real attack path.

One way to solve this problem is to increase the size of the GBF until we get a reasonable maximum false-negative probability. The problem with this approach is that the filter size may become so large that it is better to carry the IP addresses themselves instead of a compact GBF. We take a different approach that takes advantage of the GBF while still reducing the header overhead.

A. Overview

To address the high false-negative rates, we introduce an improved reconstruction procedure that eliminates false negatives at the cost of a higher false-positive probability. The proposed procedure is based on the following reasoning. During the network traversal of the attack packet, subsequent routers might invert bit markings of previous routers and cause false negatives. In the previous example, during the traversal of the attack packet through the path (R_5, R_4, R_2, R_1), either R_1 or R_2 inverts a bit previously marked by R_4. As a consequence, the GBF received by the victim does not recognize R_4 during the reconstruction procedure. To avoid this drawback, we propose that along with the GBF each router sends additional information to upstream routers during the reconstruction. This information summarizes the bit markings made by the router itself and by downstream routers in the reconstruction procedure. Therefore, it is now possible to know whether a downstream router inverted a bit marked by an upstream router.

Fig. 2 depicts the improved reconstruction procedure. For simplicity, let $k_0 = k_1 = 1$. First, the victim V recognizes R_1 in the GBF of the received attack packet and, therefore, V sends the GBF to R_1 (1). Along with the GBF, V also sends two other bit arrays of the same size, m_0 and m_1, initialized to all zeros as shown in (a). Router R_1 then updates m_0 and m_1 according to the interface from which the reconstruction request comes. Accordingly, the bits reset by that interface during the packet-marking procedure are set to 1 in m_0 and the bits set by that interface during the marking procedure are set to 1 in m_1. The updated arrays are found in (b). Later on, R_1 performs membership tests with each of its neighbors. The neighbor checking procedure slightly changes due to the additional information passed. Whenever a neighbor fails the test, R_1 does not directly discard the neighbor. Instead, R_1 checks first if it inverted a bit marking of this neighbor during the traversal of the packet through the network. To accomplish this checking, the additional bit arrays are inspected. Accordingly, R_1 checks if the inverted bits of this neighbor are set in either m_0 or m_1, depending if the neighbor's bit is 1 or 0, respectively. If the neighbor's inverted bits were overwritten, the neighbor is integrated into the attack path and discarded otherwise. In the sequence, router R_1 recognizes R_2. Router R_1 then sends the GBF along with the two updated bit arrays to R_2 (2). In its turn, router R_2 updates these arrays according to the interface from which the request comes, as shown in (c). After that, router R_2 tests the membership of R_4 in the GBF. In this case, a bit of R_4 is inverted: a bit that should be 0 is in fact 1. Nevertheless, this bit is also set in m_1, which means that it was overwritten by either R_1 or R_2 during the traversal of the packet. Therefore, router R_4 is no longer a false negative and it is integrated into the attack path. Router R_2 then sends the GBF and the two additional bit arrays indicating the bit markings made by itself and by R_1 (3). Next, router R_4 performs the same procedure with its neighbor R_5 (4). We can see in (d) the bit arrays updated by R_4. In this case, it is easy to see that the bit set in both m_0 and m_1 was reset by R_4 and overwritten later by R_2 during the traversal of the packet through the network.

An important advantage of the improved reconstruction procedure is that false negatives do not happen anymore. Since overwritten bits now can be checked at each hop, actually traversed routers are always integrated into the attack path. Therefore, the actual attack path is *always* present in

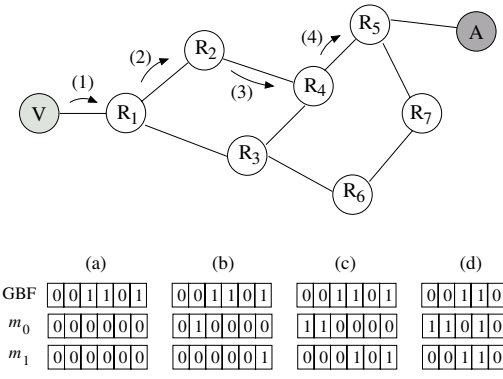

Fig. 2. The improved reconstruction procedure.

the reconstructed attack graph. On the other hand, the false-positive probability increases as a router is tested further from the victim and closer to the attacker. It happens because the fraction of bits set in m_0 and m_1 increases with the number of integrated routers. Accordingly, upstream routers are recognized as components of the attack path with higher probability. With an increasing false-positive probability, other routers are also identified as possible attacking routers and it is harder to distinguish the actual one. Nonetheless, simulation results presented in Section V show that such identifications is possible with well-chosen system parameters.

B. False Positives

The false-positive probability of the improved reconstruction procedure is calculated in a direct fashion. First, we calculate the probability that a specific bit is set in m_0 at a particular router during the reconstruction. Let q_0 be the probability that a specific bit is reset and q_1 be the probability that the bit is set during an element insertion in the GBF. Analytical expressions for q_0 and q_1 are presented in [21]. Note that a bit of m_0 being set at a router does not necessarily mean being set by that router. Either a downstream router or even the router itself can set the bit. This event happens with probability q_0 since setting a bit in m_0 is equivalent to resetting a bit in the GBF. Therefore, the probability $s_0(i)$ that a specific bit of m_0 is set at a router i hops away from the victim is the probability that at least one downstream router or the router itself set the bit. Accordingly, the probability $s_0(i)$ is

$$s_0(i) = 1 - (1 - q_0)^i. \tag{1}$$

Likewise, the probability $s_1(i)$ that a specific bit of m_1 is set at a specific router i hops away from the victim is the probability that at least one downstream router or the router itself set the bit. The probability of such event is q_1 since setting a bit in m_1 is equivalent to setting a bit in the GBF. Therefore, the probability $s_1(i)$ is

$$s_1(i) = 1 - (1 - q_1)^i. \tag{2}$$

Since the same calculation can be made to every bit of both arrays, on average, a fraction of $s_0(i)$ bits is set in m_0 and a fraction of $s_1(i)$ bits is set in m_1 at a router i hops away from the victim.

Accordingly, a bit is interpreted as zero in the membership tests of this router if the bit is reset in the GBF or if it is set in both m_1 and the GBF. Let p represents the fraction of bits in zero of the GBF received by the victim (see [21] for the analytical expression of p). The fraction $t_0(i)$ of bits that are interpreted as zero in the membership tests made by a specific router i hops away from the victim is then

$$t_0(i) = p + (1 - p)s_1(i), \tag{3}$$

Similarly, the fraction $t_1(i)$ of bits interpreted as one is

$$t_1(i) = (1 - p) + p.s_0(i). \tag{4}$$

From (3) and (4), the probability of a false positive $f_p(i)$ for a specific router i hops away from the victim in the improved reconstruction procedure is calculated as

$$f_p(i) = t_0(i)^{k_0} t_1(i)^{k_1}, \tag{5}$$

where we note that we need k_0 bits considered "zeros" and k_1 bits considered "ones" to have a false positive.

C. Analytical Results

Fig. 3 depicts the false-positive probability of a GBF as a function of i, according to (5). For every curve, the worst-case initial fraction of bits set is assumed. The worst value for the initial condition is calculated in the companion paper [20], where we show that it happens when

$$p_0 = \frac{k_0}{k_0 + k_1}. \tag{6}$$

For simplicity, let $k = k_0 = k_1$. It can be seen that increasing the number of hash functions is beneficial until a certain value. In fact, for each size of bit array, an optimal value for k_0 and k_1 exists. The corresponding tradeoff is that a larger number of hash functions implies a larger fraction of bits set in m_0 and m_1 on every hop and, therefore, a higher false-positive probability. On the other hand, when using more hash functions, it is more likely that we find an inverted bit in the GBF that is not set in m_0 nor in m_1 and, therefore, a lower false-positive probability is expected.

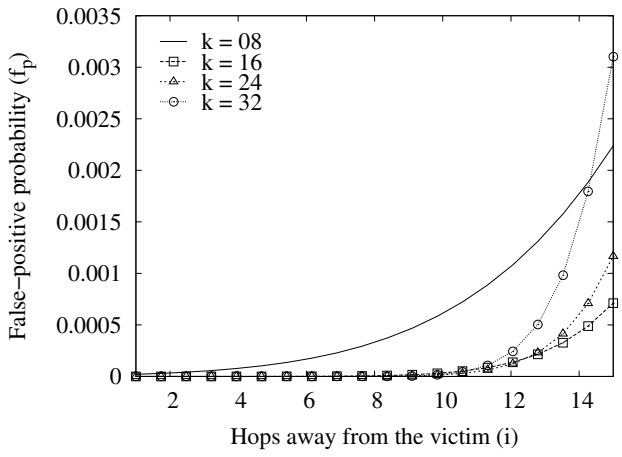

Fig. 3. False-positive probability of a GBF as a function of the distance from the victim, in hops, for $m = 256$, $n = 15$, and $p_0 = 0.5$.

Fig. 4 shows the false-positive probability of a GBF as a function of i, for different initial conditions of the filter. Accordingly, the action of the attacker increasing the false-positive probability is also limited. This constraint is caused by the same reasons that limited the attacker in the regular reconstruction procedure. For a false positive to occur, we need both bits reset and bits set and, if they are proportionally distributed as (6) states, the false-positive probability increases.

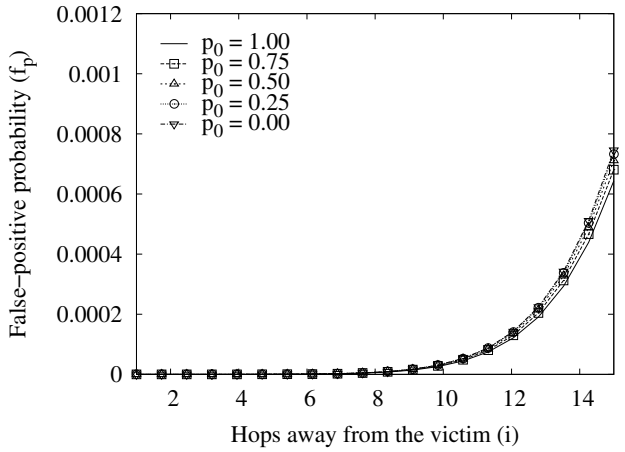

Fig. 4. False-positive probability of a GBF as a function of the distance from the victim, in hops, for $m = 256$, $n = 15$, and $k_0 = k_1 = 16$.

V. Simulation Results

In order to analyze the behavior of the improved reconstruction procedure in a Internet-based topology, we developed a C++ simulator. Additionally, we used the *nem* [22] topology generator, which is based on Internet map sampling. The *nem* generator randomly extracts a sub-graph of a network map, keeping its original properties, such as node degree, mean distance, mean eccentricity, and topology diameter. Magoni and Pansiot [22] show that the topologies generated by *nem* respect the power laws established on real Internet maps. Our topology is sampled from a real Internet map and consists of 10,000 routers. Once generated the topology, we arbitrarily choose an attacker from a set of border routers. This reflects the assumption that the core routers are not compromised and that the attacker is more likely to be in a local network out of the main Internet backbone. We then define a random loop-free attack path starting in the chosen border router. Next, we simulate the transmission of an attack packet by marking a GBF according to the routers that compose the attack path. The initial content of the GBF is always set to the worst case according to (6). Once the GBF is properly marked, the reconstruction procedure starts at the victim. Two reconstruction procedures are used. The first is the regular procedure in which only the GBF is used during the neighbor membership tests. The second is the improved reconstruction procedure described in Section IV. For every measured point, we calculated the confidence interval for a 95% confidence level, represented by vertical bars.

Fig. 5 shows the simulation results of the regular path reconstruction procedure for a typical path with 15 routers. In the figure, the length of the reconstructed path is shown as a function of the number of hash functions used in the GBF. For simplicity, we consider the same number of hash functions that set and reset bits; accordingly, $k = k_0 = k_1$. From the results, it can be seen that the regular reconstruction procedure presents a reconstructed path much smaller than the original path. For attack paths of 15 routers, only 7 routers were approximately traced. This behavior is a result of having false negatives in the reconstruction procedure. In the case of a false negative, an actually traversed router is ignored and the reconstruction procedure is prematurely interrupted. To reduce the probability of a false negative, however, a significant increase in the size of the filter is necessary, which increases the per-packet overhead. Additionally from Fig. 5, it can also be seen that by increasing k the size of the reconstructed path decreases. It happens because the false-negative probability increases with the number of hash functions. Accordingly, the probability of interrupting the reconstruction procedure at a router closer to the victim is even higher.

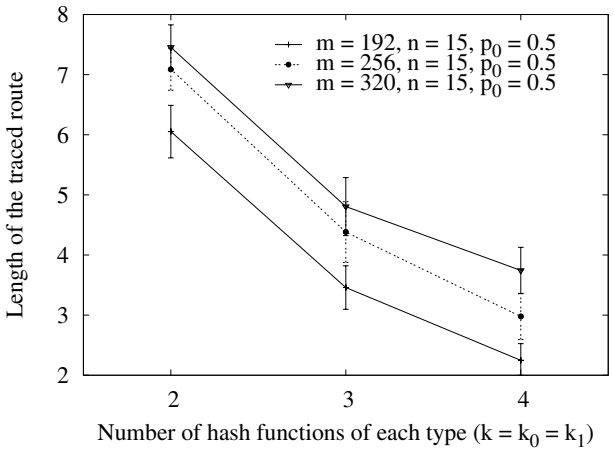

Fig. 5. Length of the reconstructed path as a function of the number of hash functions used, using the regular path reconstruction procedure.

The improved reconstruction procedure of Section IV do not present false negatives and, as a consequence, the actual attack path is always found. Nevertheless, other paths leading to innocent routers are also found due to false positives. Therefore, in order to evaluate the performance of the proposed procedure, the mean number of traced attackers is used as metric. In theory, the path reconstruction procedure should lead to only one attacker, the real one.

Fig. 6 shows the mean number of attackers traced by the improved reconstruction procedure as a function of the number of hash functions used. From the figure, it can be realized that the number of traced attackers is reduced to less than four for the 192-bit filter. Further, this number is very close to the ideal case, where only the real attack path is found, if larger filter sizes are employed. In addition, a tradeoff between the number of hash functions and the number of traced attackers can be noticed in Fig. 6. We can see that for small values of k the number of attackers is large. The same happens if

the value of k is too large. Such tradeoff is a direct result of having false positives in the reconstruction procedure. With few hash functions, few bits set and reset need to be found for a false positive to occur. On the other hand, with too many hash functions, the fraction of bits set in m_0 and m_1 increases on each hop of the reconstructed path and causes a higher false-positive probability. In both cases, a larger number of false positives leads to more routers being recognized as components of the attack path and, as a consequence, a higher number of traced attackers.

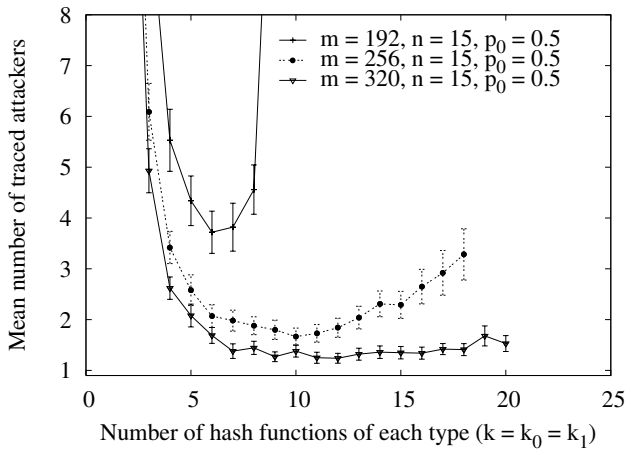

Fig. 6. Candidate attackers traced by the improved reconstruction procedure as a function of the number of hash functions used.

The number of traced attackers as a function of the length of the attack path is seen in Fig. 7. It can be noticed that the number of traced attackers increases with the length of the attack path. It happens because the false-positive probability increases as routers are tested further from the victim. As a consequence, more routers are recognized as components of the attack path and more attackers are found. With only 192 bits, we get only 3.5 candidate attackers. Using only 192 bits instead of 480 bits represents a space saving of 60% in the packet header if we were to store the complete route. We can still find a number of traced attackers closer to unity depending on the filter size.

VI. Related Work

Savage *et al.* [12] introduce an auditing-based system, where the required traceback information is located at the victim. In summary, routers overload the Identification field of the IP header to notify the victim of their presence in the attack path. Every router probabilistically inserts partial information about itself in the packets routed to the victim. After receiving enough attack packets, the victim can reconstruct the entire route. To reduce router overhead and required per-packet space, sampling and encoding techniques are employed. Although innovative, this proposal requires high computational effort during reconstruction and generates several false positives even in small-scale distributed attacks [16].

Bellovin [14] proposes a similar system for IP traceback. Whenever routing a packet, routers probabilistically send to the victim an ICMP packet with information about themselves

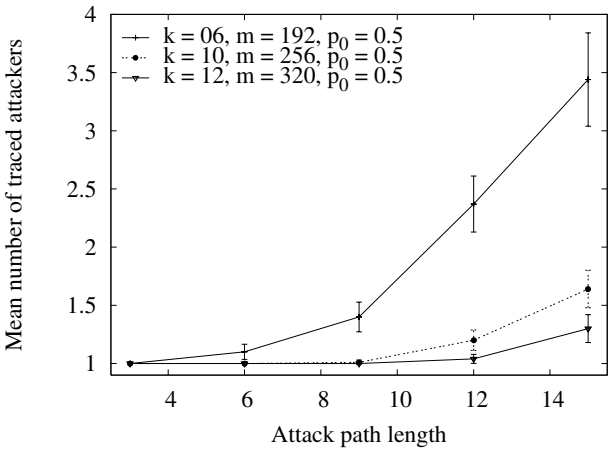

Fig. 7. Candidate attackers traced by the improved reconstruction procedure as a function of the length of the attack path.

and their adjacent routers. For a long packet flow, the victim can use the received data to reconstruct the attack path. Nevertheless, since auditing information is sent in additional router-generated packets, the attacker can send spoofed ICMP packets to disrupt path reconstruction. Therefore, messages must be authenticated to avoid spoofing. In this case, a public-key infrastructure (PKI) is necessary for victims to authenticate the out-of-band packets sent by the routers.

Dean *et al.* [13] considers probabilistic and algebraic techniques to trace IP packets. Their basic idea is that each packet carries a result of a well-known polynomial, whose unknown variables are the router IP addresses. If the victim receives enough packets from the same route, an equation system can be derived and solved with unique solution. Different from the system proposed by Savage *et al.* [12], however, this system presents no error detection code to reduce the probability of accidentally deriving an equation system by combining equations from different paths. As a consequence, even more false positives are expected to occur for small-scale distributed attacks.

Yaar *et al.* [23] propose a path identification scheme to filter DoS packets. The basic idea is that routers mark the forwarded packets with a very small "signature" of 1 or 2 bits. As the packets traverse the network, a common signature is marked on the packets that take the same path. It is worthy noticing that classifying packets with regard to their traversed paths and identifying the real source of a packet are two different problems. Although effective in distinguishing packets that take the same path, this system was not designed to trace packets back to their origin. As a result, all the victim can do is only filter the attack traffic and wait for the attack to cease. Additionally, since upstream routers can not be identified, filtering can only be made at the victim or at the nearby routers.

Another approach consists of storing auditing information in the network infrastructure. In order to reduce the amount of stored information in routers, Bloom Filters [19] can be used. Recently, these filters have been widely used in computer networks [24], [25]. Snoeren *et al.* [17] propose a scheme

that traces an attack from a single IP packet. In addition, the backtracing is done without storing all routed traffic. Instead, routers store only packet *digests* in Bloom Filters. Periodically, saturated filters are stored for future queries and replaced by new ones. To later determine if a packet traversed the router, its filter is simply checked. A recursive procedure is executed by each router to reconstruct the packet path to its true origin. The only disadvantage of such system is to keep state in the network core.

Most of the above-mentioned packet marking schemes rely on overloading the limited 16-bit Identification field of the IP header to carry path information. Although these systems impose no additional overhead to packets, they are not able to trace single-packet attacks. In order to determine the source of a single packet without storage in the network core, the whole path information must be contained in the packet itself. As a result, it is unlikely that only 16 bits are enough to accurately identify the packet's source. Our approach is different since it allows tracing a single packet back to its source and does not require per-packet state in the network core. The tradeoff cost is the additional per-packet overhead required to carry the complete path information.

VII. Conclusion

In this paper, we present an innovative approach to packet-marking IP traceback that is convenient for high-speed networks. The proposed system is able to trace an attack back to its approximate source by analyzing a single packet. Thus, our approach scales and fits well to trace sources of distributed DoS attacks. Additionally, our scheme is said to be stateless since no traceback information is storaged in the network infrastructure.

When traversing the network, packets are marked with node digests instead of full IP addresses. Upon receiving a packet, the victim disposes of a representation of the entire attack path. We introduced the Generalized Bloom Filter (GBF) to store the IP address of traversed routers. In contrast to the standard Bloom Filter, the GBF has an upper-bounded false-positive probability. The tradeoff cost is the introduction of false negatives in membership queries. False negatives are harmful because they may prematurely interrupt the reconstruction procedure. In our simulations using 15-hop routes, only 7 routers were approximately identified due to false negatives. The effect of false negatives, however, is also upper bounded and depends only on the number of hash functions, k_0 and k_1, and on the m/n ratio.

A few interesting tradeoffs could be observed with the simulation results of the improved reconstruction procedure. First, we show that there is an optimal number of hash functions that minimizes the false-positive probability and therefore the mean number of traced attackers. Besides, there is an interesting relation between the size of the filter and the size of the route being traced. For larger routes or higher accuracy, a larger filter is needed whereas for smaller routes or lower accuracy a smaller filter is enough. For instance, using $k_0 = k_1 = 6$ and 192 bits for a 15-hop route, we were able to reduce the number of candidate attackers to only 3.5 nodes.

References

[1] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson, *2006 CSI/FBI Computer Crime and Security Survey*, 2006.

[2] H. Burch and B. Cheswick, "Tracing Anonymous Packets to their Approximate Source," in *USENIX LISA'00*, New Orleans, Lousiana, USA, Dec. 2000, pp. 319–327.

[3] R. Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods," in *9th USENIX Security Symposium*, Denver, Colorado, USA, Aug. 2000, pp. 199–212.

[4] InformationWeek.com, *Dutch Botnet Bigger Than Expected*, Oct. 2005.

[5] *CERT Advisory CA-1996-26 Denial-of-Service Attack via ping*, CERT, Dec. 1996.

[6] *CERT Advisory CA-1997-28 IP Denial-of-Service Attacks*, CERT, Dec. 1997.

[7] *Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 Packets*, Cisco Systems, Inc., July 2003.

[8] F. Gont, "ICMP Attacks Against TCP," *Internet Draft: draft-gont-tcpm-icmp-attacks-03.txt*, Dec. 2004.

[9] *Vulnerability Note VU#222750: TCP/IP Implementations Do Not Adequately Validate ICMP Error Messages*, US-CERT, Apr. 2005.

[10] *CERT Incident Note IN-99-07 Distributed Denial-of-Service Tools*, CERT, Nov. 1999.

[11] Y. Zhang and V. Paxson, "Detecting Stepping Stones," in *Proceedings of the 9th USENIX Security Symposium*, Denver, Colorado, USA, Aug. 2000, pp. 171–184.

[12] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network Support for IP Traceback," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 226–237, June 2001.

[13] D. Dean, M. Franklin, and A. Stubblefield, "An Algebraic Approach to IP Traceback," *ACM Transactions on Information and System Security*, vol. 5, no. 2, pp. 119–137, May 2002.

[14] S. M. Bellovin, M. D. Leech, and T. Taylor, "ICMP Traceback Messages," *Internet Draft: draft-ietf-itrace-04.txt*, Aug. 2003.

[15] A. Mankin, D. Massey, C.-L. Wu, S. F. Wu, and L. Zhang, "On Design and Evaluation of 'Intention-Driven' ICMP Traceback," in *Proceedings of the IEEE ICCCN 2001 Conference*, Scottsdale, Arizona, USA, Oct. 2001.

[16] D. X. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," in *Proceedings of the IEEE INFOCOM 2001 Conference*, Anchorage, Alaska, USA, Apr. 2001.

[17] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer, "Single-Packet IP Traceback," *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 721–734, Dec. 2002.

[18] L. H. M. K. Costa, S. Fdida, and O. C. M. B. Duarte, "Incremental Service Deployment Using the Hop-By-Hop Multicast Routing Protocol," *IEEE/ACM Transactions on Networking*, vol. 14, no. 3, pp. 543–556, 2006.

[19] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 7, no. 13, pp. 442–426, July 1970.

[20] R. P. Laufer, P. B. Velloso, D. de O. Cunha, I. M. Moraes, M. D. D. Bicudo, and O. C. M. B. Duarte, "A New IP Traceback System against Denial-of-Service Attacks," in *12th International Conference on Telecommunications*, Capetown, South Africa, May 2005.

[21] R. P. Laufer, P. B. Velloso, and O. C. M. B. Duarte, "Generalized Bloom Filters," Universidade Federal do Rio de Janeiro, COPPE/PEE, Tech. Rep. GTA-05-43, Sept. 2005.

[22] D. Magoni and J.-J. Pansiot, "Internet Topology Modeler Based on Map Sampling," in *IEEE International Symposium on Computer Communications*, July 2002.

[23] A. Yaar, A. Perrig, and D. Song, "Pi: A Path Identification Mechanism to Defend against DDoS Attacks," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, USA, May 2003, pp. 93–107.

[24] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood, "Fast Hash Table Lookup Using Extended Bloom Filter: an Aid to Network Processing," in *Proceedings of the ACM SIGCOMM'05 Conference*, Philadelphia, PA, USA, Aug. 2005, pp. 181–192.

[25] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, "Beyond Bloom Filters: From Approximate Membership Checks to Approximate State Machines," in *Proceedings of the ACM SIGCOMM'06 Conference*, Pisa, Italy, Sept. 2006, pp. 315–326.