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, CA, 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.

1 Introduction

Denial-of-service (DoS) attacks are currently the fifth major cause of financial loss due to cybercrime [13]. 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 [4, 26]. Currently, distributed DoS (DDoS) attacks composed of 1.5 million computers are already a reality [15]. 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 [6, 7, 9, 12, 27]. 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.

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 [8]. 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 [29], 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 [22]. 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, 4, 11, 20, 22, 26]. 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 [24] 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 carrying the complete information about its path or have each router storing information about every forwarded packet [23, 26]. 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* net-

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

works. 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 [10]. 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 [3] 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 [17], we sketched the initial idea of our design and primary results were derived. In this paper, we extend our previous work by providing a detailed analytical and simulation-based evaluation of the Generalized Bloom Filter, showing that both the false-positive and false-negative probabilities are upper bounded. In addition, we also propose in this paper an improved path reconstruction procedure that eliminates the false negatives introduced by the Generalized Bloom Filter. Finally, we 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. Section 2 explains the IP traceback problem and the basic assumptions. The proposed IP traceback system is then introduced in Section 3. The analysis of our generalization of Bloom Filters is explained in Section 4. Analytical and simulation results of the Generalized Bloom Filter are presented in Section 5. The improved reconstruction procedure is introduced in Section 6. We present simulation results of the new reconstruction procedure in an Internet-based topology in Section 7. Section 8 presents the related work and conclusions are finally presented in Section 9.

2 IP Traceback

In this section, we briefly explain the definition of the IP traceback problem as introduced by Savage *et al.* [22] and outline the basic assumptions of our design.

2.1 Definitions

Figure 1 shows an example network as seen from a victim V. Every attacker A_i is a leaf node that generates attack traffic towards the victim, denoted by the dotted line. Internal nodes R_i represent routers along a path between the victim V and an attacker A_i. The set of routers R_i is also referred as the *upstream routers* from V throughout this paper. An *attack path* is an ordered list of routers between an attacker A_i and the victim V. For example, in Figure 1 there is one attack path denoted as (R_5, R_2, R_1). By grafting together the attack paths of every attacker, an *attack graph* is composed. Additionally, a router is named a *false positive* if it is in the reconstructed attack graph but it is not in the actual attack graph. Similarly, a router is named a *false negative* if it is not in the reconstructed attack graph but it is in the real attack graph.

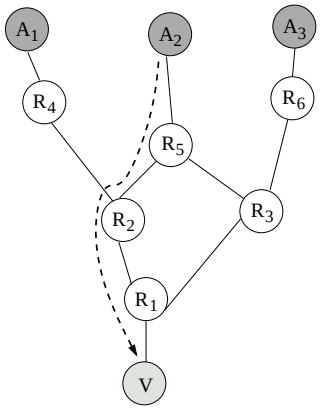

Figure 1: Network as seen from the victim V.

According to the definition, the exact traceback problem is to accurately determine the attack path from a given attacker to the victim. The exact traceback problem, however, is a hard problem [22] and therefore a more restricted problem is also postulated. The approximate traceback problem is defined as finding an attack path that contains the actual attack path as a suffix. For instance, (R_6, R_5, R_2, R_1) is a valid solution to the approximate traceback problem since it contains the actual attack path (R_5, R_2, R_1) as a suffix. Further, a solution is named robust if the attacker cannot prevent the victim from finding out an attack path with the actual attack path as a suffix.

2.2 Assumptions

Some basic assumptions are made prior to the design of the traceback system in order to establish practical guidelines and constraints:

1. attacks may consist of a single packet;
2. attackers may generate arbitrary packets;
3. attackers are aware of the traceback scheme;
4. attackers may act together;
5. routers may be compromised, but not frequently;
6. routers are resource constrained;
7. packet size should not grow as the packet traverses the network.

Previous and existing single-packet DoS attacks reinforce the first assumption [9, 12, 27]. We have covered the different ways of conducting single-packet attacks in Section 1. We emphasize here that single-packet attacks are a real threat and harder to trace than flooding attacks.

Assumptions 2-5 reflect potential capabilities of motivated attackers. First, experienced attackers are able to inject arbitrary packets into the network. Therefore, tracing systems should never rely on the initial content of packets. Secondly, the security of traceback systems must not depend on its nondisclosure. In fact, these systems should be open in order to achieve wide deployment. Third, multiple attackers with common goals may cooperate in DDoS attacks to achieve better results. Alternatively, one router may have under its control a large network of zombies which can be activated at any given time to attack a common victim. Finally, the attacker may gain access to routers by several means, but we assume this is not a frequent event. If it happens, however, the router break-in should be addressed immediately after detection to enable the complete traceback of the attack path.

Assumption 6 deals with router capabilities. We assume that the network infrastructure is resource constrained and unable to maintain per-packet state. In fact, we let the storage activity optionally to the victim while keeping low per-packet processing overhead on routers.

The final assumption that packets should not grow when traversing the network is needed to avoid fragmentation and additional processing overhead. Packet fragmentation affects network performance because additional overhead is required on the router performing the fragmentation and on the subsequent routers that carry the extra packages. Measurement evidence indicates that one significant cause of fragmentations is the insertion of the additional 40-byte tunneling header [5]. In addition, appending data to packets in flight is a resource-consuming process to routers. Ideal traceback systems therefore should not increase the packet size as it traverses the network.

3 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 [3] is used to reduce the required per-packet space and fix the size of the information inserted into the packet. 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. Figure 2 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.

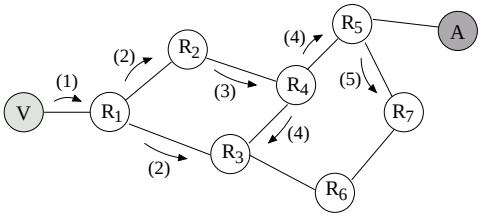

Figure 2: Example of the path reconstruction procedure.

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 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 [2, 11, 22–24]. 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.

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. We show that with 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 4, the GBF is described and an analysis of the false-positive and false-negative probabilities is derived to show the effectiveness of this new approach.

4 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. 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.

4.1 False Positives

The false-positive probability of a GBF is calculated in a similar way to the standard filter. Nevertheless, we need

first to calculate the probability of a bit being set or reset by each element insertion. Given that in a collision the functions g_i always take precedence over the functions h_j, the probability q_0 that a specific bit is reset by an element insertion is the probability that at least one of the k_0 hash functions reset the bit. Accordingly, q_0 is

$$q_0 = \left[1 - \left(1 - \frac{1}{m}\right)^{k_0}\right] \approx \left(1 - e^{-k_0/m}\right). \quad (1)$$

The probability q_1 that a specific bit is set by an element insertion is the probability that at least one of the k_1 hash functions set the bit and none of the k_0 hash functions reset the bit. Thus, q_1 is defined as

$$q_1 = \left[1 - \left(1 - \frac{1}{m}\right)^{k_1}\right]\left(1 - \frac{1}{m}\right)^{k_0} \approx \left(1 - e^{-k_1/m}\right) e^{-k_0/m}. \quad (2)$$

Finally, the probability that a specific bit remains untouched (i.e., not set nor reset) during an insertion is then

$$(1 - q_0 - q_1) = \left(1 - \frac{1}{m}\right)^{k_0+k_1} \approx e^{-(k_0+k_1)/m}. \quad (3)$$

Since the same calculation can be made for every bit in the array, on average, a fraction of q_0 bits is reset, a fraction of q_1 bits is set, and a fraction of $(1 - q_0 - q_1)$ bits remains untouched on each element insertion. For an array of m bits, we have on average $b_0 = m.q_0$ bits reset, $b_1 = m.q_1$ bits set, and $(m - b_0 - b_1)$ bits untouched on each insertion.

From these probabilities, the distribution of bits over the bit array can be determined. The probability p that a specific bit is 0 after n insertions is calculated from the probabilities of $n + 1$ mutually exclusive events. The first event is when the bit is initially 0 and remains untouched by the n elements. If p_0 represents the probability that a specific bit is initially reset, the probability of such event is $p_0 (1 - q_0 - q_1)^n$. The next n events are those where the bit is reset by the $(n - i)$-th element and remains untouched by the following i elements, for $0 \leq i \leq n - 1$. The probability of each one of these events is $q_0 (1 - q_0 - q_1)^i$. Accordingly, we have

$$p = p_0 (1 - q_0 - q_1)^n + \sum_{i=0}^{n-1} q_0 (1 - q_0 - q_1)^i = p_0 (1 - q_0 - q_1)^n + \frac{q_0}{q_0 + q_1}[1 - (1 - q_0 - q_1)^n]. \quad (4)$$

Since the same computation can be made for every bit in the array, on average, a fraction of p bits is reset and a fraction of $(1 - p)$ bits is set after n insertions.

From the bit-array distribution, the probability of a false positive can be easily calculated. Since on average b_0 bits are reset and b_1 bits are set on each element insertion, the probability of a false positive f_p for the GBF is calculated as

$$f_p = p^{b_0}(1 - p)^{b_1}. \quad (5)$$

4.2 False Negatives

False positives happen for external elements with the same probability for each checked element. On the other hand, false negatives occur only for *inserted elements* with a different probability for each element. One factor that directly affects the false-negative probability is the insertion order. For instance, first inserted elements have higher chances of being false negatives than last inserted elements. It happens because the first elements have more elements inserted after them, and therefore the probability of inverting one of their marked bits is higher.

The false-negative probability can be calculated if we have the probability that a specific bit from the $(n-i)$-th element is not inverted by the subsequent i elements, for $0 \leq i \leq n-1$. The probability $p_{00}(n-i)$ that a bit reset by the $(n - i)$-th element remains in 0 by the end of the following i insertions is calculated from the probabilities of $i + 1$ mutually exclusive events. The first event is when the bit remains untouched by all of the subsequent i insertions; it happens with probability $(1 - q_0 - q_1)^i$. The other i events are those where the bit is reset by the $(n - j)$-th element and remains untouched throughout the following j insertions, for $0 \leq j \leq i - 1$. Therefore, $p_{00}(n - i)$ is defined as

$$p_{00}(n-i) = (1 - q_0 - q_1)^i + \sum_{j=0}^{i-1} q_0 (1 - q_0 - q_1)^j = (1 - q_0 - q_1)^i + \frac{q_0}{q_0 + q_1}\left[1 - (1 - q_0 - q_1)^i\right]. \quad (6)$$

In the same way, the probability $p_{11}(n - i)$ that a bit set by the $(n - i)$-th element remains in 1 by the end of the following i insertions is

$$p_{11}(n-i) = (1 - q_0 - q_1)^i + \sum_{j=0}^{i-1} q_1 (1 - q_0 - q_1)^j = (1 - q_0 - q_1)^i + \frac{q_1}{q_0 + q_1}\left[1 - (1 - q_0 - q_1)^i\right]. \quad (7)$$

From Equations (6) and (7), the false-negative probability of the inserted elements can be calculated. The false-negative probability $f_n(n - i)$ of the $(n - i)$-th element is calculated by taking the complement of the probability that none of its bits are inverted. Accordingly, this probability is

$$f_n(n-i) = 1 - p_{00}(n-i)^{b_0} p_{11}(n-i)^{b_1}. \quad (8)$$

4.3 Application

In the proposed IP traceback system, the elements inserted into the packet GBF 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 allocated 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.

5 Results

In order to analyze the behavior of the GBF, we implemented a simulator using C++ [16]. The simulator is based on a class called GBF that contains the methods for inserting and checking elements in a bit array. For each simulation round, we select new hash functions, set the bit array to a given initial condition, and insert the elements into the filter. After the insertions, membership queries of external elements and inserted elements are performed to respectively measure the false-positive and false-negative rates. For the independent and random hash functions assumed in Section 4, we used in our simulations a universal class of hash functions [21]. This class is defined as follows. Let the elements be integers drawn from a universe $S = \{1, 2, \ldots, z-1\}$, and let the output range of the hash functions be $\{0, 1, \ldots, m-1\}$. The class H of hash transformations $h_{c,d}$ which map an element $x \in S$ into the respective range is

$$H = \{h_{c,d}(\cdot) \mid 0 < c < z,\ 0 \leq d < z\}, \qquad (9)$$

where

$$h_{c,d}(x) = [(cx + d) \bmod z] \bmod m. \qquad (10)$$

The numbers c and d are integers within the interval defined by Equation (9). For each hash function, c and d are arbitrarily chosen. The integer z is defined as a large prime.

Additionally, to show the advantages of the GBF over the standard filter, we also present an analytical comparison between the two versions of the filter in this section. Our goal is to demonstrate the necessity and advantages of the GBF over the standard filter for representing the traversed route. The analysis includes two different metrics: false positives and false negatives.

We must emphasize that the simulation results were extremely close to the analytical results, which corroborates the analytical expressions derived in the previous section. In our simulations, for a 95% confidence level, the largest confidence interval obtained was only 0.003. It means that, besides matching the analytical results, the simulation results have also a very small variance. Since this confidence interval is too small, it is not presented in the following graphs. In this section, all graphs present simulation results as discrete points and analytical results as continuous curves.

5.1 Upper-bounded False Positives

During the path reconstruction procedure, a false positive implies the integration of an incorrect router into the attack path. As the false-positive probability increases, inaccurate routers and reconstructed paths are identified. Accordingly, false positives difficult distinguishing the attacker from other innocent routers. Moreover, they also add unnecessary processing overhead in false-positive routers. Therefore, a low false-positive probability is desired.

First, we note that Equations (4) and (5) can be simplified if we assume that $m \gg k_0$ and $m \gg k_1$. This assumption is quite reasonable since usually the size of the filter is much larger than the number of hash functions used. In this case, we can rewrite Equation (4) as follows

$$p = p_0 \left(1 - q_0 - q_1\right)^n + \frac{k_0}{k_0 + k_1}\left[1 - \left(1 - q_0 - q_1\right)^n\right]. \qquad (11)$$

In addition, we can also rewrite $b_0 \approx k_0$ and $b_1 \approx k_1$. Accordingly, the simplified probability of a false positive using these assumptions is

$$f_p \approx p^{\,k_0}(1-p)^{k_1}. \qquad (12)$$

Figure 3 shows the false-positive probability of a GBF f_p as a function of $(1-p)$, according to Equations (11), (12), and our simulation results. The probability $(1-p)$ can also be seen as the fraction of bits set after inserting n elements. For the standard Bloom filter, we can see that the false-positive probability increases with $(1-p)$.

A clear tradeoff between the distribution of bits and the false-positive probability can be noticed in the curves representing the GBF. This tradeoff can be understood by observing that, on average, k_0 bits reset and k_1 bits set are required to cause a false positive. If, however, $(1-p)$ is low, it is easy to find a bit reset but it is hard to find a bit set. On the other hand, if $(1-p)$ is high, it is easy to find a bit set and difficult to find a bit reset.

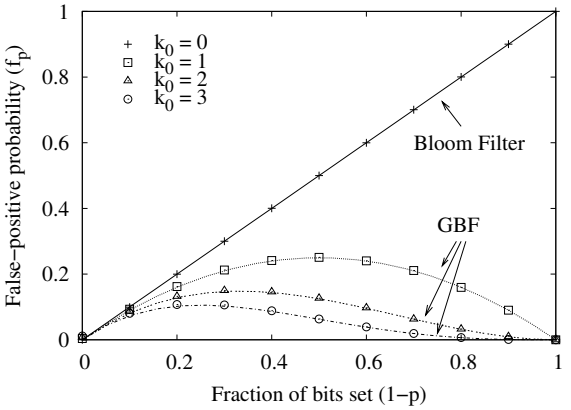

Figure 3: False-positive probability of a GBF as a function of the fraction of bits set, for $k_1 = 1$.

By differentiating Equation (12) with respect to p, it is easy to check that the maximum false-positive probability of a GBF is reached when

$$p = \frac{k_0}{k_0 + k_1}, \tag{13}$$

and it can be shown that this is indeed a global maximum. The maximum false-positive probability F_p is determined by substituting the result of Equation (13) back into Equation (12)

$$F_p = \left(\frac{k_0}{k_0 + k_1}\right)^{k_0} \left(\frac{k_1}{k_0 + k_1}\right)^{k_1}. \tag{14}$$

Different from the standard Bloom Filter, the GBF has a bounded false-positive probability F_p exclusively determined by the number of hash functions used, k_0 and k_1. For instance, when using as few hash functions as $k_0 = k_1 = 2$ we have a maximum false-positive probability of only 6.3%. If we increase the number of hash functions to $k_0 = k_1 = 3$, the maximum false-positive probability drops to 1.6%. This probability can be further reduced if a larger number of hash functions is used. The tradeoff cost is an increase in the false-negative probability, as explained in the next subsection.

The inherent property of having an upper bound in the false-positive probability is exactly what we want. Since this upper bound does not depend on the fraction of bits set in the array, there is no way that an attacker can generate filters with false-positive rates higher than F_p. The effect of this simple attack can be successful mitigated just by using a GBF instead of a standard filter in the packet header to register the traversed routers.

5.2 Upper-bounded False Negatives

When using a GBF in the traceback system, false negatives might occur during the path reconstruction procedure. A false negative implies not detecting an actually traversed router. It is worth mentioning, however, that the attacker do not interfere with the false-negative probability, according to Equations (6), (7) and (8). It can be seen that the term p_0 do not appear on these equations.

Figure 4 shows the false-negative probability for each element $(n-i)$, for $0 \leq i \leq n-1$, according to Equation (8) and our simulation results, using the parameters $k_1 = 1, m = 1280$ and $n = 10$. According to our previous definitions, Element 1 represents the first inserted element and Element 10 is the last inserted element. First, we can notice that the false-negative probability always equals zero for the standard Bloom Filter, as expected. Since the original Bloom Filter does not use hash functions to reset bits, it is impossible to have false negatives. For the GBF, however, we can notice that elements inserted first have higher false-negative probabilities. It happens because first elements have more elements inserted after them and, as a consequence, the probability of inverting their bit markings is higher. Another important observation is that the false-negative probability increases with k_0. It happens because the more functions we use, the higher is the probability of an element to have one of its marked bits inverted by a subsequent element. The result is similar if k_1 increases instead (not shown). Accordingly, the number of hash functions used in the GBF has a dual effect. On one hand, increasing the number of hash functions reduces the false-positive probability, as shown in Section 5.1. On the other hand, it increases the false-negative probability.

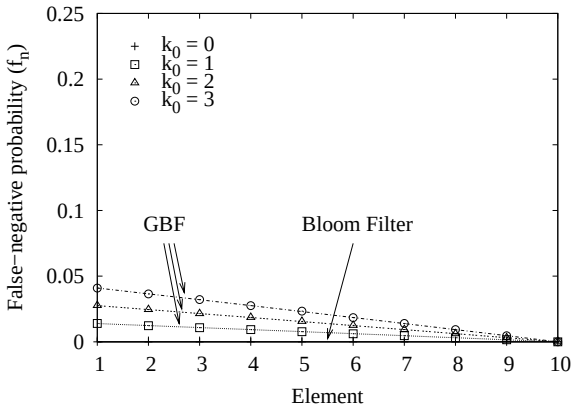

Figure 4: False-negative probability of a GBF for each inserted element, using $k_1 = 1, m = 1280, n = 10$.

According to Figure 4, the false-negative probability is a monotonically decreasing function of the number of

inserted elements. In fact, this behavior is always true and it can be analytically proven if we observe that, for two elements x and y with x inserted before y, we have $p_{00}(x) \leq p_{00}(y)$ and $p_{11}(x) \leq p_{11}(y)$. From Equation (8), it is trivial to check then that $f_n(x) \geq f_n(y)$ is always true. Therefore, we can derive an upper bound on the false-negative probability of a GBF. Let $f_n(0)$ be the false-negative probability of an hypothetical element inserted prior to the n elements of a given set. Following our previous result, we can say that the inequality $f_n(0) \geq f_n(1) \geq f_n(2) \geq \ldots \geq f_n(n)$ always holds. Accordingly, we can define the maximum false-negative probability of a GBF as $f_n(0)$. An advantage in doing so is that we can represent the maximum false-negative probability as a function of only k_0, k_1, and the m/n ratio.

Assuming $m \gg k_0$ and $m \gg k_1$, we first rewrite Equations (6) and (7) for the hypothetical element 0 as

$$\begin{aligned} p_{00}(0) &= \\ &= (1 - q_0 - q_1)^n + \frac{k_0}{k_0 + k_1}\left[1 - (1 - q_0 - q_1)^n\right] \\ &= e^{-(k_0+k_1)n/m} + \frac{k_0}{k_0 + k_1}\left(1 - e^{-(k_0+k_1)n/m}\right), \end{aligned} \tag{15}$$

$$\begin{aligned} p_{11}(0) &= \\ &= (1 - q_0 - q_1)^n + \frac{k_1}{k_0 + k_1}\left[1 - (1 - q_0 - q_1)^n\right] \\ &= e^{-(k_0+k_1)n/m} + \frac{k_1}{k_0 + k_1}\left(1 - e^{-(k_0+k_1)n/m}\right). \end{aligned} \tag{16}$$

Substituting Equations (15) and (16) back into (8), and noticing that $b_0 \approx k_0$ and $b_1 \approx k_1$, the maximum false-negative probability F_n is defined as

$$F_n = f_n(0) \approx 1 - p_{00}(0)^{k_0} p_{11}(0)^{k_1}. \tag{17}$$

According to Equations (15), (16), and (17), F_n is uniquely defined by k_0, k_1, and the m/n ratio. Therefore, the system designer may first arbitrate the desired maximum false-positive probability of the GBF by defining both k_0 and k_1. The desired maximum false-negative probability may then be achieved by adjusting the m/n ratio. Lower false-negative probabilities, however, are achieved at the cost of using more bits per element.

Figure 5 depicts the maximum false-negative probability of a GBF as a function of the m/n ratio, according to Equation (17) and simulation results. From the figure, we can see that increasing m/n leads to a lower false-negative probability. It occurs because by increasing the number of bits per element we increase the output range of the hash functions, decreasing the probability of a bit overwriting.

According to the results mentioned so far, the GBF is capable of representing a set with limited false positives and limited false negatives, regardless of the state of the bit array. For instance, when using $k_0 = k_1 = 2$ and $m/n = 128$ bits per element, the GBF yields no more than 6.3% false positives and no more than 6.0% false negatives. If lower rates are desired, we can use $k_0 = 2$, $k_1 = 3$, and $m/n = 256$ bits per element. In this case, we have a maximum false-positive probability of 3.5% and a maximum false-negative probability of 4.6%. False negatives can be further reduced if we use $k_0 = 2$, $k_1 = 3$, and $m/n = 512$ bits per element. In that case, we get a maximum false-negative probability of only 2.3%. The value of these parameters can be further increased as long as we are willing to reduce the false-positive and false-negative probabilities.

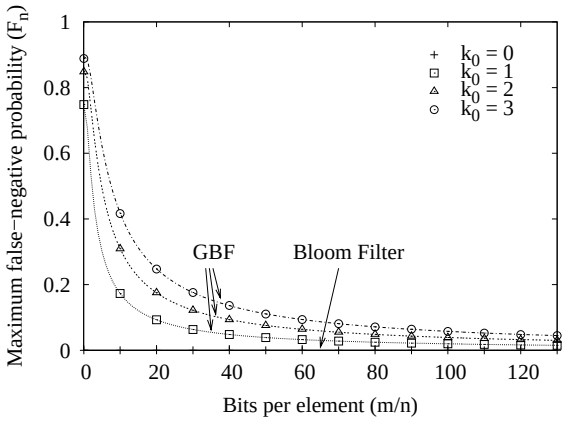

Figure 5: False-negative probability of a GBF as a function of the number of bits per element, using $k_1 = 1$.

6 Improved Reconstruction Procedure

While using a GBF limits the action of the attacker in generating false positives, it also introduces false negatives 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 Figure 2. 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.

6.1 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.

Figure 6 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 the bit that is 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.

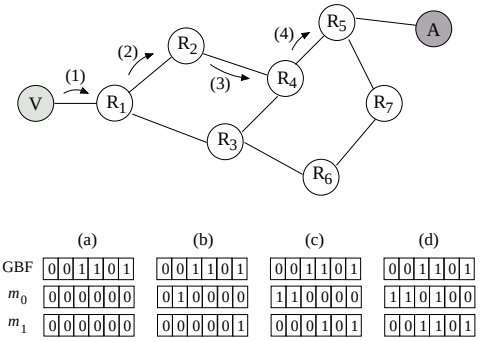

Figure 6: The improved reconstruction procedure.

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 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 7 show that such identifications is possible with well-chosen system parameters.

6.2 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. Note that a bit 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

$$s_0(i) = 1 - (1 - q_0)^i . \tag{18}$$

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

$$s_1(i) = 1 - (1 - q_1)^i . \quad (19)$$

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. Therefore, 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

$$t_0(i) = p + (1 - p)s_1(i). \quad (20)$$

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

$$t_1(i) = (1 - p) + p.s_0(i). \quad (21)$$

From Equations (20) and (21), 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

$$f_p(i) = t_0(i)^{b_0} t_1(i)^{b_1}, \quad (22)$$

where b_0 and b_1 are respectively the average number of bits reset and bits set needed for a false positive to occur, as described in Section 4.

6.3 Analytical Results

Figure 7 depicts the false-positive probability of a GBF as a function of i, according to Equation (22). 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 [17], where we show that it happens when

$$p_0 = \frac{k_0}{k_0 + k_1}. \quad (23)$$

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.

Figure 8 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

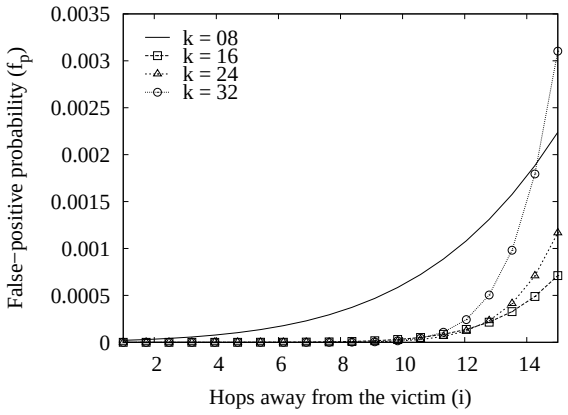

Figure 7: False-positive probability of a GBF as a function of the distance from the victim, in hops, for $m = 256$, $n = 15$, and $p_1(0) = 0.5$.

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 Equation (13) states, the false-positive probability increases.

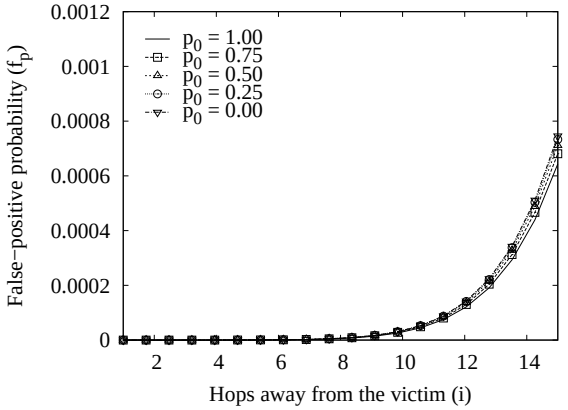

Figure 8: False-positive probability of a GBF as a function of the distance from the victim, in hops, for $m = 256$, $n = 15$, and $k_0 = k_1 = 16$.

7 Simulation Results

In order to analyze the behavior of the improved reconstruction procedure in a Internet-based topology, we developed a C++ simulator [16]. Additionally, we used the *nem* [19] 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 [19] 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 Equation (23). 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 6. For every measured point, we calculated the confidence interval for a 95% confidence level, represented by vertical bars.

Figure 9 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 Figure 9, 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.

The improved reconstruction procedure of Section 6 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 actual one.

Figure 10 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.

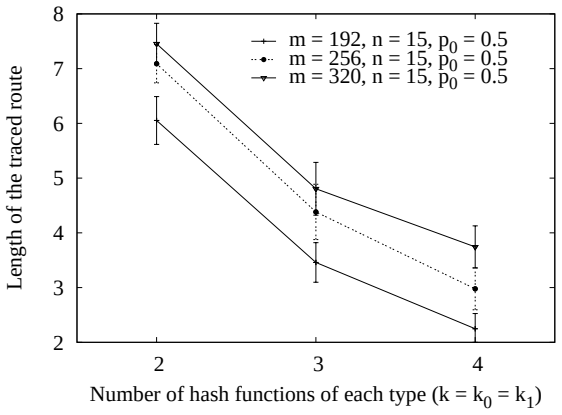

Figure 9: Length of the reconstructed path as a function of the number of hash functions used, using the regular path reconstruction procedure.

In addition, a tradeoff between the number of hash functions and the number of traced attackers can be noticed in Figure 10. 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.

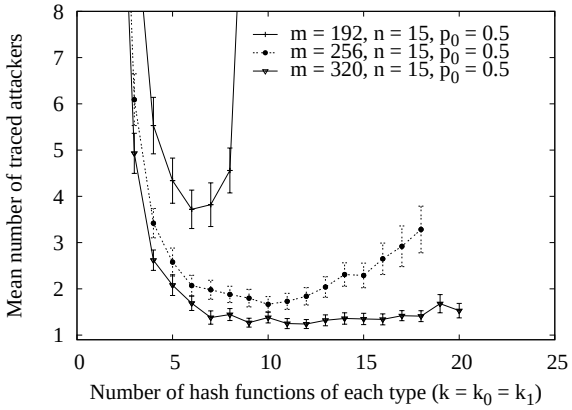

Figure 10: Candidate attackers traced by the improved reconstruction procedure as a function of the number of hash functions used.

The number of traced attackers as a function of the length of the attack path is seen in Figure 11. 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.

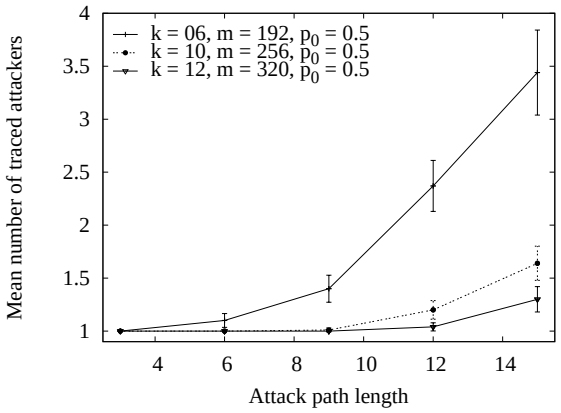

Figure 11: Candidate attackers traced by the improved reconstruction procedure as a function of the length of the attack path.

Figure 12 shows the mean number of traced attackers as a function of the m/n ratio of the GBF, for different path lengths. In this figure, we see how header overhead can be traded off for accuracy. The improved reconstruction procedure always traces the real attacker even for long paths, but the accuracy decreases as we use fewer bits per element. It happens because false positives increase as we reduce the filter size. According to Internet statistics [14], the distribution of path lengths has a mean value of 15.3 and a standard deviation of 4.2. Supposing a gaussian distribution, the probability of a path length being less than or equal to 24 is approximately 98%. As a result, we can see from the curve where $n = 24$ that the system can trace 98% of Internet routes with an accuracy of 5.3 using only 12 bits per element; that is, 62.5% less header overhead than if we were to save a 32-bit IPv4 address and 90.6% less if we were to save an 128-bit IPv6 address. If we use more bits, we can even improve the accuracy of the reconstruction procedure. For a given m/n ratio, we can also see that smaller routes benefit from higher accuracy levels.

8 Related Work

Savage *et al.* [22] 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

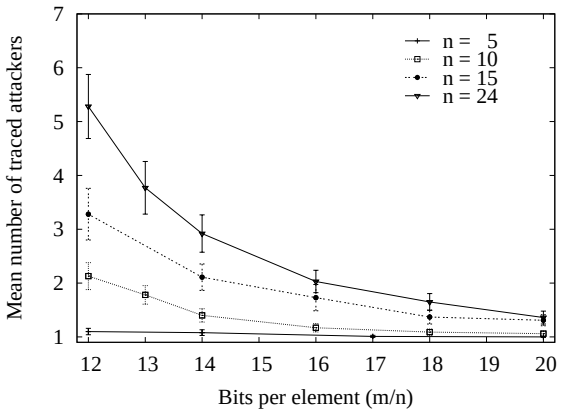

Figure 12: Candidate attackers traced by the improved reconstruction procedure as a function of the number of bits per element.

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 [24].

Bellovin [2] proposes a similar system for IP traceback. Whenever routing a packet, routers probabilistically send to the victim an ICMP packet with information about themselves 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 is necessary for victims to authenticate the out-of-band packets sent by the routers.

Dean *et al.* [11] 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.* [22], 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.* [28] 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 sig-

nature 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 completely 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. As a consequence, network resources are wasted to unnecessarily carry the attack traffic towards the victim.

Another approach consists of storing auditing information in the network infrastructure. The simplest way to collect auditing trails is to log every traversed packet in routers [26]. Although quite simple, this approach requires excessive resources for both data storage and data mining. For instance, storing every packet of a saturated OC-24 (1.244 Gbps) link requires the exorbitant amount of 9.3 GB per minute or, equivalently, 13.4 TB per day. In addition, a compromised router may cause privacy problems since it contains information about every routed packet.

An alternative to reduce the amount of stored information is to use Bloom Filters [3]. Recently, these filters have been widely used in computer networks [1, 25]. Snoeren *et al.* [23] 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. Improvements proposed by Li *et al.* [18] drastically reduce the space required for data storage in the core even though the capability of tracing a single packet is compromised.

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 trade-off cost is the additional per-packet overhead required to carry the complete path information.

9 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. We show that the false-positive probability of a GBF is upper bounded and this bound depends uniquely on the number of hash functions used, k_0 and k_1. For instance, by using as few hash functions as $k_0 = k_1 = 3$, the maximum false-positive probability of a GBF is only 1.6%. The tradeoff cost is the introduction of false negatives. False negatives are harmful because they may prematurely interrupt the reconstruction procedure. Nonetheless, we show in this paper that the effect of false negatives is also upper bounded and depends only on the number of hash functions, k_0 and k_1, and on the m/n ratio. As a result, the designer may first arbitrate the maximum false-positive probability accepted and fix k_0 and k_1. The maximum false-negative probability can then be achieved by tuning m/n value.

An improved reconstruction procedure is also introduced to eliminate false negatives at the cost of a little increase in false positives. Through simulations, our improved reconstruction procedure is proven to be both effective and accurate in an Internet-based topology. 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.

Acknowledgments

The authors would like to thank Luis Henrique Costa, Deborah Estrin, and Lixia Zhang for their remarkable and valuable comments.

References

[1] 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.

[2] BELLOVIN, S. M., LEECH, M. D., AND TAYLOR, T. ICMP Traceback Messages. *Internet Draft: draft-ietf-itrace-04.txt* (Aug. 2003).

[3] BLOOM, B. H. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM 7*, 13 (July 1970), 442–426.

[4] BURCH, H., AND CHESWICK, B. Tracing Anonymous Packets to their Approximate Source. In *USENIX LISA'00* (New Orleans, Lousiana, USA, Dec. 2000), pp. 319–327.

[5] CAIDA. *Packet Fragmentation on the UCSD-CERF Link*, June 2002.

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

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

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

[9] CISCO SYSTEMS, INC. *Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 Packets*, July 2003.

[10] COSTA, L. H. M. K., FDIDA, S., AND DUARTE, O. C. M. B. Incremental Service Deployment Using the Hop-By-Hop Multicast Routing Protocol. *IEEE/ACM Transactions on Networking 14*, 3 (2006), 543–556.

[11] DEAN, D., FRANKLIN, M., AND STUBBLEFIELD, A. An Algebraic Approach to IP Traceback. *ACM Transactions on Information and System Security 5*, 2 (May 2002), 119–137.

[12] GONT, F. ICMP Attacks Against TCP. *Internet Draft: draft-gont-tcpm-icmp-attacks-03.txt* (Dec. 2004).

[13] GORDON, L. A., LOEB, M. P., LUCYSHYN, W., AND RICHARDSON, R. *2006 CSI/FBI Computer Crime and Security Survey*, 2006.

[14] HUFFAKER, B., FOMENKOV, M., PLUMMER, D. J., MOORE, D., AND CLAFFY, K. Distance Metrics in the Internet. In *Proceedings of the IEEE International Telecommunications Symposium* (2002).

[15] INFORMATIONWEEK.COM. *Dutch Botnet Bigger Than Expected*, Oct. 2005.

[16] LAUFER, R. P., VELLOSO, P. B., AND DE O. CUNHA, D. GBF Traceback Simulator. Available: http://www.gta.ufrj.br/~rlaufer/gbf_traceback.

[17] LAUFER, R. P., VELLOSO, P. B., DE O. CUNHA, D., MORAES, I. M., BICUDO, M. D. D., AND DUARTE, O. C. M. B. A New IP Traceback System against Denial-of-Service Attacks. In *12th International Conference on Telecommunications* (Capetown, South Africa, May 2005).

[18] LI, J., SUNG, M., XU, J., AND LI, L. Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Theoretical Foundation. In *Proc. of the 25th IEEE Symposium on Security and Privacy* (Oakland, California, USA, May 2004).

[19] MAGONI, D., AND PANSIOT, J.-J. Internet Topology Modeler Based on Map Sampling. In *IEEE International Symposium on Computer Communications* (July 2002).

[20] MANKIN, A., MASSEY, D., WU, C.-L., WU, S. F., AND ZHANG, L. On Design and Evaluation of 'Intention-Driven' ICMP Traceback. In *Proceedings of the IEEE ICCCN 2001 Conference* (Scottsdale, Arizona, USA, Oct. 2001).

[21] RAMAKRISHNA, M. V. Practical Performance of Bloom Filters and Parallel Free-Text Searching. *Communications of the ACM 32*, 10 (Oct. 1989), 1237–1239.

[22] SAVAGE, S., WETHERALL, D., KARLIN, A., AND ANDERSON, T. Network Support for IP Traceback. *IEEE/ACM Transactions on Networking 9*, 3 (June 2001), 226–237.

[23] SNOEREN, A. C., PARTRIDGE, C., SANCHEZ, L. A., JONES, C. E., TCHAKOUNTIO, F., SCHWARTZ, B., KENT, S. T., AND STRAYER, W. T. Single-Packet IP Traceback. *IEEE/ACM Transactions on Networking 10*, 6 (Dec. 2002), 721–734.

[24] SONG, D. X., AND PERRIG, A. Advanced and Authenticated Marking Schemes for IP Traceback. In *Proceedings of the IEEE INFOCOM 2001 Conference* (Anchorage, Alaska, USA, Apr. 2001).

[25] SONG, H., DHARMAPURIKAR, S., TURNER, J., AND LOCKWOOD, J. 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.

[26] STONE, R. CenterTrack: An IP Overlay Network for Tracking DoS Floods. In *9th USENIX Security Symposium* (Denver, Colorado, USA, Aug. 2000), pp. 199–212.

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

[28] YAAR, A., PERRIG, A., AND SONG, D. 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.

[29] ZHANG, Y., AND PAXSON, V. Detecting Stepping Stones. In *Proceedings of the 9th USENIX Security Symposium* (Denver, Colorado, USA, Aug. 2000), pp. 171–184.