Secure Split Assignment Trajectory Sampling: A Malicious Router Detection System

Sihyung Lee, Tina Wong, Hyong S. Kim

June 9, 2006
CMU-CyLab-06-009

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Secure Split Assignment Trajectory Sampling: A Malicious Router Detection System

Sihyung Lee Tina Wong Hyong S. Kim

ECE Department and CyLab

Carnegie Mellon University

sihyunglee@cmu.edu, tinaw@andrew.cmu.edu, kim@ece.cmu.edu

Abstract

Routing infrastructure plays a vital role in the Internet, and attacks on routers can be damaging in numerous ways. Compromised routers can drop, modify, mis-forward or reorder valid packets. However, existing proposals for secure forwarding require substantial computational overhead and additional capabilities at routers. We propose *Secure Split Assignment Trajectory Sampling (SATS)*, a system that detects malicious routers on the data plane. SATS locates a set of suspicious routers when packets do not follow their predicted paths. SATS works with a traffic measurement platform using packet sampling, has low overhead on routers and is applicable to high-speed networks. Different subsets of packets are sampled over different groups of routers (called *Split Range Assignment*) to ensure attackers cannot completely evade detection. Our evaluation shows that SATS can significantly limit a malicious router's harm to a small portion of traffic in a network.

I. Introduction

Routers are crucial to the Internet. Unfortunately, attacks aimed directly at routers are prevalent and on the rise. According to CERT/CC, there are lists of thousands of compromised routers being traded underground [1]. Such lists contain not only access routers, but also core routers; some are Cisco GSRs with BGP peers in external ASs. There are hacker tools, openly available on the Web, to scan, identify and eventually exploit routers with weak passwords and default settings. It took only 32 hours to scan an entire class A of 2^{24} IP addresses for vulnerable routers [2].

More recently, Lynn [3] presented Cisco IOS's security flaws, which allow hackers to seize control of Cisco's Internet routers.

Compromised routers are being used as platforms to send spam, launch DoS attacks, intercept sensitive traffic, and carry out illegal yet profitable activities. For example, a malicious BGP capable router can redirect traffic for spammers with a "dump-and-run" attack, by announcing and thereby stealing unused address space belonging to a legitimate organization [4]. Another example is to allow infiltration into an enterprise's VPN by changing configurations and security policies at a border router between enterprise and its provider. In general, since routers are considered trusted entities in a network, their power can be easily exploited once they are compromised.

In order to secure the Internet routing infrastructure, the three main planes of network functionality (i.e. management, control and data) must be protected. The management plane deals with the configuration and management of network elements. Protecting this plane involves more than using secure connections to access routers. The NSA has published guidelines [5] on how to harden a router's configuration. The NSA recommends to shut down insecure services like telnetd and unnecessary ones like httpd, and provides examples of defining access control lists to block intruders. While every router should be hardened with a security template, there is the issue of time-of-check versus time-of-use: the security template is usually applied once when a router is initially configured, or only during infrequent network audits. Also, the static analysis of configuration files cannot detect Byzantine behavior, especially if a router's operating system has been tampered with.

The control plane runs intradomain and interdomain routing protocols to build forwarding tables at routers. On the other hand, the data plane forwards (or drops) packets according to forwarding tables built by the control plane. Recently, considerable research and industrial efforts have addressed securing routing protocols. S-BGP [6] and the IETF RPSEC working group [7] are two examples. A secure version of BGP provides path and prefix attestations, which prevent propagation of illegitimate routes. Even in the presence of a secure control plane, however, a compromised router can disregard decisions made by the control plane and act autonomously and maliciously on the data plane. It can modify, drop, delay, reorder, mis-forward valid packets or permit otherwise prohibited packets. Such misbehavior would not be prevented by any secure routing protocol.

The related work on detecting data-plane misbehavior falls into two categories: active probing and passive monitoring. Secure Traceroute [8] is one of the active probing approaches. It actively sends probing packets to detect packet

drops, modifications and mis-forwardings. Once a human operator notices performance degradation on a path, Secure Traceroute is initiated. Secure Traceroute-capable routers then respond to the probing packets. Stealth Probing [9] sends probing packets through an encrypted channel with normal packets. The normal packets are thus indistinguishable from the probing packets. To protect all the packets, every packet has to be encrypted. The active probing methods require the probing packets to be similar to the packets that will be attacked. However, it is not easy to determine beforehand which packets will be targeted by an attacker. Furthermore, it is not clear when to initiate probing and which routes to probe. In this paper, we focus on passive monitoring.

The Conservation of Flow (CF) [10] is one of the passive monitoring approaches. CF analyzes traffic volume at various observation points in a network. Discrepancies between the ingress traffic volume and the egress traffic volume at different points indicate potential problems. CF can discover dropping of packets but fails to detect modification of packets if traffic volume remains the same. Hughes et al. [10] also addresses several ways to fool the CF algorithm. Fatih [11] considers other types of attacks including modification, substitution, mis-forwarding and reordering attacks. Routers process more information, such as hashes of packet content, to validate traffic content and content order in addition to the traffic volume. To reduce the overhead on routers, Fatih proposes a path-level detection algorithm while increasing the size of the suspicious set of routers. In both CF and Fatih, most of detection processing is done on routers.

This paper presents Secure Split Assignment Trajectory (SATS), a system that detects packet modification, substitution, dropping and reordering attacks as well as routing loops carried out by subverted routers. SATS observes paths that packets have followed and detects malicious attacks if the observed paths are not consistent with the predicted paths of the packets. A set of suspicious routers are then detected. Consecutive malicious routers are also detected if there is at least one correct router at the boundary of the set of malicious routers. SATS is designed to detect malicious routers in high-speed networks where we can only examine a subset of packets due to limited processing capacity. The sampled packets are then further aggregated. While sampling can decrease the monitoring overhead at each router in a high-speed network, the accuracy of detection depends on how samples are selected. If the samples do not contain compromised packets, the attacks cannot be detected. Thus, SATS proposes a method called *Split Range Assignment* that prevents the samples from being biased or tampered with by attackers. In order to sample packets and to observe paths that the sampled packets have followed, SATS utilizes a traffic measurement system, Trajectory

Sampling [12], and secure communication between measurement systems and routers. SATS has a low impact on router processing and memory usage. It only requires a modular hash function on each packet and a cryptographic hash function on selected packets. The rest of the processing is done externally independent of routers.

Caveats: We limit our focus to routers within a single administrative domain, such as ISP, university campus or enterprise network, or multiple cooperating administrative domains which are open to share measurement data with one another. Extending SATS to work across uncooperative networks is left as future work. We also assume that the network topology of a domain is known. SATS cannot detect mis-forwarding behavior that does not manifest itself as a forwarding loop.

This paper is organized as follows. We first introduce Trajectory Sampling in Section II, which can be used for detecting malicious elements. In Section III, we illustrate the design of SATS. In Section IV, we show the capability of SATS for more intelligent malicious routers. We evaluate the detection rate, incremental deployment property, and full path completion time of SATS in Section V. We then conclude in Section VI.

II. Background

In this section, we briefly describe previous works on flow-level measurement and Trajectory Sampling (TS). We discuss only material relevant to SATS. The flow-level measurement provides more fine-grained information about network traffic than the traditional link-level SNMP approach. A router periodically sends records about its flows to a collection machine that processes the records. Cisco's Netflow [13] as well as flow-level measurement solutions from other vendors are widely deployed in networks. These tools are used to calculate traffic matrices to provision a network, understand traffic mix in terms of application types, and find reasons behind sudden traffic spikes.

For high data rate interfaces, packet sampling is necessary to scale flow-level measurement. Otherwise, a router quickly runs out of processing cycles and memory while trying to examine every passing packet, and its forwarding performance is severely degraded. Netflow uses a simple 1-out-of-N sampling method but not without problems [14]. Improved methods have been proposed, such as the novel Trajectory Sampling (TS) proposed by Duffield and Grossgaluser [12], [15], [16]. The main idea of TS is that a packet is either sampled at every router along its path, or not sampled at all. It works as follows:

- A router applies a *selection hash function*, $h_{selection}()$, to compute a *hash value* over the invariant portion of a packet. The source and destination IP addresses, port numbers, protocol and payload remain the same as a

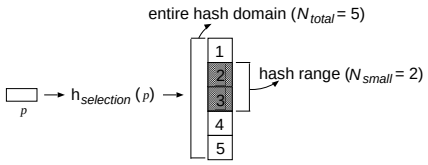


Fig. 1. An example of a selection hash function and hash range. The selection hash function has a finite number of output values, {1,2,3,4,5} ($N_{total} = 5$). Only when the hash value of a packet falls in the hash range, {2,3}, the packet is sampled ($N_{small} = 2$).

packet travels across the network, and thus are included in the calculation. On the other hand, the TTL, ECN, TOS and CRC checksum are not included in the calculation as they can be changed.

In order to achieve unbiased and uniform sampling, this hash function must generate values that appear statistically independent of its input. Using large packet traces from a Tier-1 ISP network, Duffield and Grossglauser show that modular arithmetic with prime moduli satisfies this property. Modular arithmetic can be implemented using simple integer arithmetic in hardware or software. [12] shows that current technology can compute such a function for each packet that arrives at 20 Gbps or even higher.

- If the hash value falls into the predetermined *hash range*, the packet is sampled. The size of this hash range, N_{small} divided by the total size of the *hash domain*, N_{total}, is the effective sampling rate, p_{samp}. Fig. 1 shows an example of a selection hash function and the hash range. The selection hash function has a total of 5 output values, {1,2,3,4,5} ($N_{total} = 5$). Among the 5 output values, 2 values, {2,3}, are selected for the hash range ($N_{small} = 2$). Thus the sampling rate $p_{samp} = N_{small}/N_{total} = 2/5$. The actual sampling rates used in high-speed networks are 1/100, 1/400, 1/1000 and smaller.
- For each sampled packet, a *label hash function*, $h_{label}()$, computes another hash value over the invariant portion of a packet. The hash value is called the *label* of the packet since the label provides a unique ID of the packet. This label is then reported back to a machine for processing – this is called the *backend engine* in TS. A different hash function is used to make the label small, yet unique during a measurement interval. An ingress router also reports a *key*, which contains raw header information of a flow, for each sampled packet.
- The backend engine reconstructs the path a sampled packet traversed in the network. This path is called the packet's *trajectory*. The backend engine sets a timer when it first receives a label l. When the timer expires, the backend engine gathers the routers that sent label l and reconstructs the packet's trajectory. It assumes that the

network topology is known. A possible timeout value is the sum of the upper bounds of all possible delays: the delay within the label buffer before a label is exported, and the propagation and queuing delay a packet experiences from a reporting router to the backend engine. The backend engine also provides storage, query and visualization functions.

- The selection hash function, hash range and label hash function must be identical at all routers during a measurement interval. This ensures that a packet is sampled at all routers on its path or at none. If each router randomly samples a subset of packets, as in Netflow's 1-in-N method, a packet's path cannot be reconstructed. To adjust the sampling rate, an out-of-band mechanism is used to change the hash range on all routers.

Besides the benefits of fine-grained flow-level measurements and reduction of overhead on router performance through sampling, TS has a number of additional advantages. It is a direct observation method: a packet's or flow's path is measured directly, without needing to know IGP and BGP routing state. On the other hand, Netflow-style data requires routing state, such as routing table dumps or protocol updates, in order to derive the original path of a packet or flow. This indirect method is not as accurate as the direct observation, as there are synchronization issues during the computation. Another advantage is that the reconstructed trajectories can be used to passively measure link performance, without the injection of active probe traffic into a network. The label reports can include timestamps of when the packet was sampled, and with these timestamps, the packet delays and loss rates between routers can be derived.

The IETF is working on standardizing various aspects of flow-level measurements including TS to ensure multi-vendor compatibility and industry-wide acceptance. For example, the IP Flow Information Export (IPFIX) working group [17] is chartered to define the process of exporting flow information from IP devices. The Packet Sampling (PSAMP) working group [18] aims to specify a framework, protocol and different selection techniques for network elements to select subsets of packets for measurement-based purposes.

III. Design of SATS

A. Overview of SATS

Fig. 2 illustrates the components of SATS, which consists of the following functions.

Split Range Assignment: SATS assigns multiple overlapping hash ranges to routers to minimize the probability that an attacker can completely evade detection. In contrast, TS uses a single universal hash range for every router in a network,

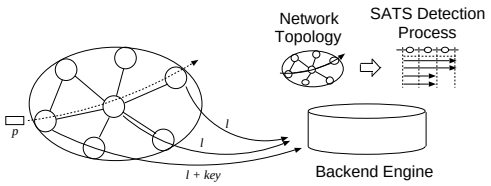


Fig. 2. Overview of SATS. A packet p is sampled at every router on its path. The sampled labels, l's, along with the key are then reported to the backend engine. The trajectory of p is reconstructed in the backend engine and fed into the SATS detection process. The SATS detection process then aggregates the trajectory according to the flow of p and looks for inconsistencies in the aggregation to detect malicious attacks.

which is vulnerable to attacks. We call our scheme *Split Range Assignment* and TS's *Single Range Assignment*.

Report Collection: A router samples a subset of packets based on its assigned hash ranges and reports the hash labels and keys of sampled packets to the backend engine. SATS is designed with router performance in mind, and we aim to keep router state and processing to a minimum. The next three functions are carried out by the backend engine, not by routers.

Reconstruction and Aggregation of Trajectories: At the end of each measurement interval, the backend engine reconstructs trajectories of sampled packets using reports from routers. Trajectories with the same ingress router and destination routing prefix pair are in turn aggregated. SATS detects anomalies based on this aggregation unit instead of a single trajectory.

Detection of Inconsistent Trajectories: In each aggregation, SATS looks for inconsistent trajectories, which are different from their predicted trajectories.

Pinpointing Malicious Routers: If inconsistent trajectories are found, SATS locates a set of suspicious routers that are responsible for the attack.

In the next section, we first describe Split Range Assignment in detail.

B. Split Range Assignment

Let us consider the case of a malicious router that not only misbehaves but also tries to avoid detection by SATS. One way to avoid detection is to attack packets that are not being sampled. In TS's Single Range Assignment, the malicious router knows that the hash range it has been assigned is the only hash range being sampled throughout the network. In our proposed Split Range Assignment, we vary hash ranges from router to router. Different hash ranges are assigned by the backend engine through encrypted and authenticated channels to ensure that the hash range assigned to a router is unknown to other routers. Thus, only the backend engine has knowledge of the entire

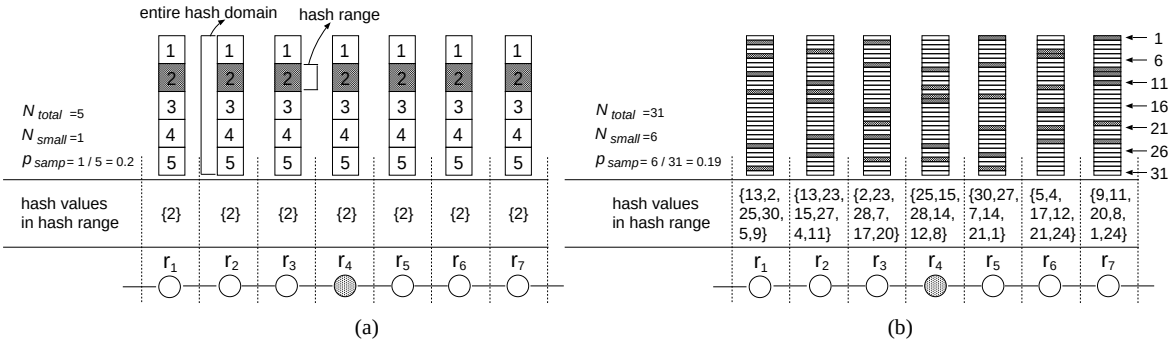


Fig. 3. Hash ranges in Single Range Assignment (a) and Split Range Assignment (b).

range assignments. PSAMP [19] specifies the secure configuration and exportation of data as a requirement for the packet selection and reporting.

Fig. 3 shows the concept of Split Range Assignment. A long vertical rectangle at a node represents the entire hash domain of the selection hash function. Within the entire hash domain, the small gray rectangles depict the hash values in the hash range at the node. As opposed to Single Range Assignment in Fig. 3.(a), Split Range Assignment in Fig. 3.(b) has hash ranges that varies from a router to a router. However, the sampling rate $p_{samp} = N_{small}/N_{total} = 6/31 = 0.19$ at a router remains approximately the same as that of Single Range Assignment, $p_{samp} = N_{small}/N_{total} = 1/5 = 0.20$. If a malicious router, r_4, ever attacks packets outside of its hash range $\{25, 15, 28, 14, 12, 8\}$, e.g. hash value 4, the attack will soon be noticed by $\{r_2, r_6\}$. Note that Split Range Assignment is not random sampling (as in Cisco Netflow) since more than one router would be assigned the same hash values, such as the hash value 4 in $\{r_2, r_6\}$ in Fig. 3.(b). If all the routers have different hash ranges (as with random sampling), we cannot compare samples from a router with samples from any other routers.

We assign different hash ranges as follows. A router shares one hash value with each router in a network. Thus, if there are N_{router} routers in the network, $N_{small} = N_{router} - 1$ hash values are assigned to a router. Given p_{samp}, we find N_{small} and N_{total} for the selection hash function as shown in Fig. 4. For example, with $p_{samp} = 1/5$ and $N_{router} = 7$ routers in the network (Fig. 3.(b)), $N_{total} = 31$, which is the smallest prime number greater than or equal to $(N_{router} - 1)/p_{samp} = (7 - 1) \times 5 = 30$. $N_{small} = \lfloor N_{total} \times p_{samp} \rfloor = \lfloor 31/5 \rfloor = 6$. Then $N_{small} = 6$ hash values are assigned to each router in a way that one router has one hash value in common with each of the other 6 routers. For instance, r_1 shares the hash values 13, 2, 25, 30, 5 and 9 with r_2, r_3, r_4, r_5, r_6 and r_7, respectively. We evaluate the detection rate of this assignment method in Section V-B, and we show that the probability that an attacker can

$\mathcal{S}_{total}$: a set of all the routers in a network
$\mathcal{PR}$: a set of prime numbers, where $prime_i \in \mathcal{PR}$ is the i-th element of $\mathcal{PR}$
N_{router}: the number of routers in $\mathcal{S}_{total}$

$N'_{small} = N_{router} - 1$; $\quad N'_{total} = N'_{small}/p_{samp}$;
$N_{total} = min_i(prime_i),\ prime_i \geq N'_{total}$;
$N_{small} = \lfloor (N_{total} \times p_{samp}) \rfloor$;
for each pair of routers (r_i,r_j), $i < j$, $r_i \in \mathcal{S}_{total}$, $r_j \in \mathcal{S}_{total}$:
  select one hash value h_i out of N_{total} hash values in the entire hash domain at random
  assign h_i to (r_i,r_j)
for each router r_i:
  select a set of $(N_{small} - N'_{small})$ hash values, $\mathcal{H}$, out of N_{total} hash values in the entire hash domain at random
  assign $\mathcal{H}$ to r_i

Fig. 4. The advanced version of Split Range Assignment algorithm.

avoid detection is fairly low.

In the remainder of this section, we describe the SATS detection process in detail.

C. The SATS Detection Process

Split Range Assignment prevents malicious routers from tampering with the sampling process. Sampled packets are then reported to the backend engine, where the trajectories of the packets are reconstructed. The trajectories are then fed into the SATS detection process. The SATS detection process consists of three steps and is carried out in every *measurement interval*.

Step 1: Aggregate trajectories belonging to the same *flow*, which we define to be trajectories with the same ingress router and destination routing prefix pair.

Step 2: Check for inconsistencies in each set of aggregated trajectories. Mark any inconsistency as suspicious trajectories.

Step 3: Pinpoint a set of potentially malicious routers that could be responsible for the inconsistent trajectories.

We first provide the main premise of the SATS detection process. We then elaborate each step in detail in the following sections. Again, all the steps in SATS detection process are carried out in the backend engine, not in routers.

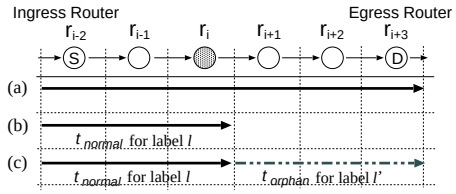


Fig. 5. The predicted trajectory of a packet p (a) and the resulting trajectories of p when p was dropped at node r_i (b) and when p was modified at node r_i (c).

C.1 Inconsistent Trajectories

The main premise of the SATS detection process is that if packets are manipulated, the resulting trajectories will not be consistent with the predicted trajectories. Fig. 5 shows the predicted and resulting trajectories of a packet p, which is supposed to go from an ingress node r_{i-2} to an egress node r_{i+3}. The label of p is $l = h_{label}(p)$. The predicted trajectory of p is shown in Fig. 5.(a). If p is dropped at a node r_i (Fig. 5.(b)), the trajectory of p, t_{normal}, ends at r_i before reaching its egress node. If p is modified to p' at r_i (Fig. 5.(c)), the trajectory for the label l, t_{normal}, also ends prematurely at r_i since the label is changed to $l' = h_{label}(p')$ before p is forwarded to r_{i+1}. In the case of modification, a new trajectory, t_{orphan}, for the new label l' starts from r_{i+1} if $h_{selection}(p')$ falls in the hash range. We refer to the new trajectory as an *orphan trajectory* as opposed to a *normal trajectory* since the new trajectory has no origination point, that is, no ingress router reported the trajectory's label, l'. When t_{orphan} is discarded, the two early ended normal trajectories, t_{normal}'s, in (b) and (c) look the same. Consequently, we use the early ending of a normal trajectory as a clue to detect both packet dropping and modification. Orphan trajectories are discarded since we already have a way to detect anomalies. Substitution of a packet for another packet yields two modified packets and thus is detected as well.

In Split Range Assignment, a packet p sampled from a hash value h is not sampled at all the routers on the path. Thus, the trajectory t of p has holes on the routers with hash values other than h. Only if t has a hole in any of the routers where h is assigned, the trajectory is inconsistent. t is an orphan trajectory if t has a hole in the first router where h is assigned on p's path. In Section V-D, we have shown that SATS can perform traffic measurement functions of TS even with holes in trajectories.

We first focus on the detection of packet dropping, modification and substitution. The detection of routing loops, reordering attacks and timing attacks is described in detail in the last subsection.

$t_i[r_j]$: an array that shows whether the trajectory i, t_i, has a report from the j-th node, r_j. If t_i has a report from r_j, $t_i[r_j] = 1$. If not, $t_i[r_j] = 0$.
$\mathcal{H}(i)$: a set of N_{small} hash values in the hash range assigned to the i-th node, r_i.
for each normal trajectory t_i sampled from a hash value h in a flow:
  for each router r_j on the path:
    if ($t_i[r_j] = 1$ and $h \in \mathcal{H}(i)$) **then** $C_{prim}[h][r_j] = C_{prim}[h][r_j] + 1$;

Fig. 6. Aggregation algorithm.

C.2 Aggregation of Multiple Trajectories

At the end of each measurement interval, trajectories are aggregated into the same flow, which is defined as the trajectories with the same ingress router and destination routing prefix pair. Based on each *aggregation* rather than each trajectory, we make a decision concerning anomalies. The aggregation is done mainly for two reasons. First, running one detection process for each set of aggregated trajectories scales better than running one detection process for each trajectory, especially in a high load situation where thousands of packets are sampled each second. Second, the aggregation helps us use a threshold to differentiate between legitimate packet drops and malicious attacks. We do not raise an alarm for each inconsistent trajectory, which might be caused by congestion or an error in the packet header[1]. Instead, we raise an alarm only when the number of inconsistent trajectories in an aggregation is more than the threshold[2].

In order to identify the flow of a normal trajectory t, we need to know both the ingress router and destination routing prefix. We use the key reported from the ingress router since the key includes the destination routing prefix. However in Split Range Assignment, t may have a hole in the ingress router. Thus, we have all routers, not only the ingress routers, report a key for each sampled packet. The key has to include the source IP and destination routing prefix. We then identify the ingress router of t through another trajectory t' that does not have a hole in its ingress router. The ingress router of t' is the same one for t if t' has the same source IP and destination prefix as those of t.

Fig. 6 shows the pseudo code of the aggregation algorithm. The aggregation process can be thought of as overlaying of trajectories one after another. In the backend engine, we maintain a *primary counter*, C_{prim}, for each hash value

[1]The loss of report packets en route to the backend engine can also create incomplete trajectories. [16] provides a way to infer the packet loss rate from the report loss rate. The report loss rate is estimated from the sequence numbers carried by report packets.

[2]Setting the threshold depends on environment. For example, legitimate packet losses in wireless networks is much more likely than in wired networks. Thus, the threshold must be determined accordingly. Also, the threshold has to be changed adaptively according to the average traffic volume. The number of samples collected from a link and the link capacity can help to predict the legitimate packet loss ratio and thus can help to determine the threshold.

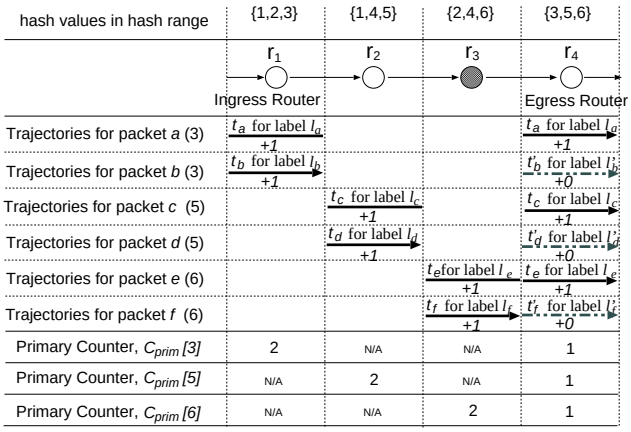


Fig. 7. Aggregation of Trajectories in Split Range Assignment. The number in the parentheses next to a trajectory is the hash value where the trajectory is sampled. The primary counter values are shown at the bottom.

assigned to each node on the path. $C_{prim}[h]$ counts the number of normal trajectories sampled from a hash value h in a node. Thus, if some of the trajectories end prematurely at a node r, C_{prim} then decreases for nodes beyond the node r in the path. If the decline is more than the threshold TH_{prim}, the aggregation is marked as an anomalous one.

Fig. 7 illustrates an example of the aggregation algorithm. We assume that the hashes of the six trajectories are distributed into three different hash values, 3, 5 and 6. A trajectory sampled from a hash value h has "holes" on the routers where h is not assigned. However, the aggregation of trajectories shows the complete path followed by the packets in the flow, which is the *predicted trajectory* of the packets. A malicious router, r_3, modifies half of the packets, b, d and f. The modified packets result in three orphan trajectories, $\{t'_b, t'_d, t'_f\}$. The orphan trajectories are then discarded during the aggregation, since we already have a way to detect anomalies as shown by the early ending of normal trajectories, $\{t_b, t_d, t_f\}$. As the normal trajectories of b, d and f, $\{t_b, t_d, t_f\}$, end at r_3, they do not increment C_{prim}'s for nodes beyond r_3. Thus when the normal trajectories are aggregated, C_{prim} decreases beyond r_3.

C_{prim} represents the number of normal trajectories sampled from each router. Note that C_{prim} is maintained in the backend engine, not in each router, and updated when trajectories are aggregated in the backend engine.

C.3 Finding Inconsistencies in Each Aggregation and Pinpointing Malicious Routers

Once trajectories are aggregated, we then check for inconsistent trajectories in each aggregation and pinpoint a set of suspicious routers. We show the algorithm in Fig. 8. In order to find inconsistent trajectories, we compare

$\mathcal{A}$: a set of aggregations, where $a_i \in \mathcal{A}$ is an individual aggregation i
$\mathcal{H}(i)$: a set of N_{small} hash values in the hash range assigned to the i-th router, r_i
$\mathcal{S}_{suspect}$: a set of suspicious routers
$C_{prim}[h_j][r_i]$: the primary counter that counts the number of normal trajectories collected from the hash value h_j of a router r_i.

find_reduction(a_i) // $a_i \in \mathcal{A}$ is the aggregation examined now
  for each node r_l in a_i:
    for each hash value $h_k \in \mathcal{H}(l)$:
      if $((C_{prim}[h_k][r_l] - C_{prim}[h_k][r_r]) > TH_{prim})$ **then** // r_r is the closest node to r_l in a_i, where $\mathcal{H}(r) \ni h_k$ and $r > l$
        $\mathcal{S}_{suspect} = \mathcal{S}_{suspect} \cup \{r_s : l \leq s \leq r\}$;

Fig. 8. Detection and Pinpointing algorithm for packet drop, modification and substitution in Split Range Assignment.

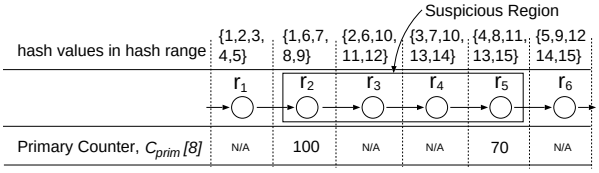


Fig. 9. Suspicious region when $C_{prim}[8]$ decreases at r_5. $\{r_2, r_3, r_4, r_5\}$ are reported as suspicious.

C_{prim}'s for two routers $\{r_i, r_j\}$ $(i < j)$ where the same hash value h is assigned. We compare the samples of the same packets. If the decrease in C_{prim} for $\{r_i, r_j\}$ is more than TH_{prim}, we pinpoint all the routers between r_i and r_j including $\{r_i, r_j\}$ to be *the suspicious region* that includes a set of routers $\mathcal{S}_{suspicious} = \{r_k : i \leq k \leq j\}$. Fig. 9 shows an example of a suspicious region in Split Range Assignment. Since r_2 and r_5 have the same hash value 8, the $C_{prim}[8]$'s for $\{r_2, r_5\}$ are compared. Since we see a decrease in the $C_{prim}[8]$'s for $\{r_2, r_5\}$, we conclude that $\{r_2, r_5\}$ and the routers between $\{r_2, r_5\}$ are suspicious. Thus $\mathcal{S}_{suspicious} = \{r_2, r_3, r_4, r_5\}$.

The reason for choosing $\mathcal{S}_{suspicious}$ as suspects is based on three possible scenarios: (In Fig. 9, r_i and r_j correspond to r_2 and r_5, respectively.)

Scenario 1: r_i is malicious. The packets were dropped at r_i. r_j reported correctly that it had not seen the packets. r_j did not report the dropped packets to the backend engine. Alternatively, when there are consecutive malicious routers, one of the previous nodes $r_l \in \{r_n : 1 \leq n < i - 1\}$ dropped the packets, the following nodes from r_{l+1} to r_{i-1} misinformed the backend engine that they had observed the packets, and finally r_i reported correctly.

Scenario 2: r_j is malicious. r_i forwarded the packets correctly, but r_j did not report the packets to the backend engine. r_j may or may not have dropped the packets.

Scenario 3: One of the routers between r_i and r_j, that is, $r_m \in \{r_n : i+1 \leq n \leq j-1\}$, is malicious. r_m dropped the packets. The hashes of the dropped packets do not fall in the hash ranges of any routers between r_m and r_j, $\mathcal{S}_{non} = \{r_n : m \leq n \leq j-1\}$. In other words, the packets under attack are not supposed to be sampled at any of the routers in $\mathcal{S}_{non}$. Thus, C_{prim}'s for $\mathcal{S}_{non}$ do not decrease. On the other hand, the hashes of the packets fall in the hash range of r_j, and $C_{prim}(r_j)$ thus decreases.

In all three scenarios, one of the routers in $\mathcal{S}_{suspicious}$ must be faulty.

Reduction of Suspicious Region: To further reduce the size of $\mathcal{S}_{suspicious}$, we assign the hash value h that is common in the hash ranges of $\{r_i,\ r_j\}$ to the router in the middle of $\{r_i,\ r_j\}$ as shown in Fig. 10. Fig. 11 illustrates an example. C_{prim} initially decreases at r_6 and $\mathcal{S}_{suspect} = \{r_2, r_3, r_4, r_5, r_6\}$ (Fig. 11.(a)). The hash value 10, which is common in (r_2, r_6), is assigned to r_4. In the next measurement interval, there can be 4 possible cases. If $C_{prim}[10]$ decreases at r_4 (Fig. 11.(b)), $\mathcal{S}_{suspect}$ is reduced to $\{r_2, r_3, r_4\}$. If $C_{prim}[10]$ decreases at r_6 (Fig. 11.(c)), $\mathcal{S}_{suspect}$ is reduced to $\{r_4, r_5, r_6\}$. In both cases, the hash value 10 is assigned to the router in the middle of $\mathcal{S}_{suspect}$ (r_3 and r_5 in (b) and (c), respectively), and we resume the detection process until the size of $\mathcal{S}_{suspect}$ is 2. The malicious router may stop attacking packets (Fig. 11.(d)). Then $\mathcal{S}_{suspect}$ of the previous measurement interval, $\{r_2, r_3, r_4, r_5, r_6\}$, is reported as suspicious. If the malicious router starts to attack packets whose hashes fall in a different hash value, the hash value 6 in Fig. 11.(e), the intersection of $\mathcal{S}_{suspect}$ and $\mathcal{S}_{suspect}$ of the previous measurement interval, $\mathcal{S}_{suspect} \cap \mathcal{S}_{suspect\_old} = \{r_s : 1 \leq s \leq 7\} \cap \{r_s : 2 \leq s \leq 6\} = \{r_s : 2 \leq s \leq 6\}$, is reported as suspicious.

Consecutive Malicious Routers: Although there might be other colluding routers before r_i as shown in scenario 1, we first examine $\mathcal{S}_{suspicious}$. Among all the consecutive colluding routers, the last one must be in $\mathcal{S}_{suspicious}$. If the routers before r_i continue to behave maliciously, they all will be eventually detected one by one. In summary, if there is a correct node at the end of the colluding routers, all colluding routers are detected.

Incremental Deployment: Between r_i and r_j, there might be a set of routers, $\mathcal{S}_{non}$, that do not have deployed SATS. In this case, the routers in $\mathcal{S}_{non}$ are also included in the suspicious region since C_{prim} decreases at the same node r_j if any one of $\mathcal{S}_{non}$ have manipulated the packets.

Routing Changes and Equal-Cost Multi-Paths: We may observe an aggregation where a node, r_i, branches into $n(>1)$ nodes, from rb_1 to rb_n, if there are equal-cost multi-paths, or routing changes. In this case, we compare C_{prim} for

```
detection() {
  for each aggregation a_i:
    S_suspect = {};
    find_reduction(a_i);
    while(size(S_suspect) > 2) {
      m = ⌊(l+r)/2⌋;  // S_suspect = {r_s : l ≤ s ≤ r}
      assign h to r_m;  // h ∈ H(l) ∩ H(r)
      S_suspect_old = S_suspect;
      find_reduction(a_i);
      if (S_suspect = φ) then {   // The malicious router stops attacking packets.
        S_suspect = S_suspect_old;
        break;
      }
      if (S_suspect ⊄ S_suspect_old) then {   // The malicious router changes target hash ranges.
        S_suspect = S_suspect ∩ S_suspect_old;
        break;
      }
    }
    report(S_suspect);
}
```

Fig. 10. Detection and Pinpointing algorithm that reduces the size of a suspicious region $\mathcal{S}_{suspect}$ in Split Range Assignment.

r_i with the sum of the C_{prim}'s for all the nodes branching from r_j. Thus, if $((C_{prim}[h][r_i] - \sum_{k=1}^{n} C_{prim}[h][rb_k]) > TH_{prim})$, we suspect the set $\{r_i\} \cup \{rb_k : 1 \leq k \leq n\}$. Then, from each of rb_k, we resume the detection process. Fig. 12 shows an example where node 4 branches into node A and node D. We assume that node 4, A, and D have the same hash value. The detection process is running at node 4. If $(100 - (30 + 40) > TH_{prim})$, both node A and D along with node 4 are reported as suspicious. If there are other routers with different hash values between {node 4, node A} or between {node 4, node D}, the routers are also included in the suspicious region.

Malicious mis-forwarding of packets can also create an aggregation branching into two or more. Malicious mis-forwarding directs packets to sub-optimal paths or to paths where the packets cannot get to their destinations. In order to detect such mis-forwardings, we need to correlate the aggregation path with the state of the routing protocol and link states. Further study is needed to obtain timely and accurate information about the routing state.

D. Extensions to SATS

So far, we have focused on the detection of packet dropping, modification and substitution. In this section, we introduce the following two extensions to SATS. These processes are also based on the sampling, reporting and trajectory reconstruction mechanisms in TS.

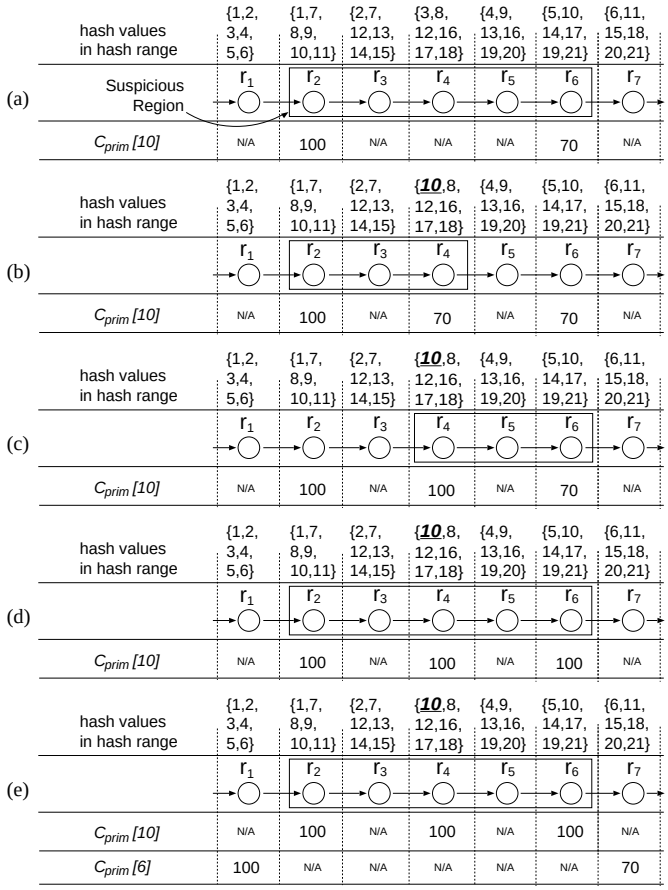


Fig. 11. (a): Suspicious region when $C_{prim}[10]$ decreases at r_6. $\{r_2, r_3, r_4, r_5, r_6\}$ are reported as suspicious. (b), (c), (d) and (e): Suspicious regions when the hash value 10 is assigned to r_4.

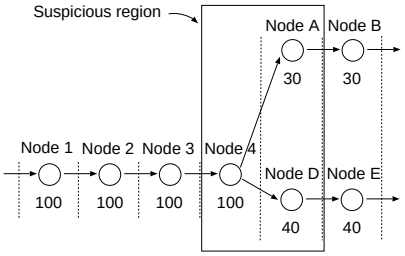


Fig. 12. Suspicious region when an aggregation branches into two paths. The number below a node represents the primary counter C_{prim} for the node.

- *Detection of Routing Loop*: If we find an aggregation of trajectories that intersects itself, i.e. if we see any repeated nodes in the path, we report it as a routing loop. We also need a threshold for the number of self-intersections in order to reduce the false positives caused by the routing loops when routing is temporarily unstable. The suspicious region would be in one router where the aggregation intersects itself.

Λ_i: labels collected from router i, r_i
Π_{ij}: $\Lambda_i \cap \Lambda_j$ ordered by the timestamps from Λ_i
$C_{reorder}[r_i]$: a counter that adds the number of labels reordered at r_i
$\mathcal{S}_{suspect}$: a set of suspicious routers

$\mathcal{S}_{suspect} = \{\}$;
for each flow i:
  for each hash range hr_j assigned to the flow i:
    for each router r_k ($k > 1$) where hr_j is assigned:
      if ($\Pi_{k(k-1)} \neq \Pi_{(k-1)k}$) **then**
        increment $C_{reorder}[r_k]$ by the number of reordered packets;
  for each router r_j ($j > 1$) of the flow i:
    if ($C_{reorder}[r_j] > TH_{reorder}$) **then**
      $\mathcal{S}_{suspect} = \mathcal{S}_{suspect} \cup \{r_{j-1}, r_j\}$;
report($\mathcal{S}_{suspect}$);

Fig. 13. Reordering detection algorithm in Split Range Assignment.

- *Detection of Packet Reordering Attacks*: The algorithm for detecting packet reordering attacks is shown in Fig. 13. For each aggregation, we can detect packet reordering attacks by comparing the time-ordered list of labels from a router with the list from another router with the same hash range. These routers collect the same labels. The labels are ordered by their timestamps in reports. The number of reordered packets is counted for each router at the backend engine. If the number for a router is more than the threshold $TH_{reorder}$, we mark the router and the upstream router as suspicious.

  Fig. 14 shows an example of reordering detection for the labels collected from link A. The time-ordered list of common labels from link A and the list from link B are the same as $\{1, 2, 3, 4\}$, which means that the packets from link B are not reordered when forwarded to router 2. On the contrary, the list from link C and the list from link A are different ($\{a,b,c,d\} \neq \{b,a,c,d\}$), meaning that packet *a* and *b* from link C are reordered when forwarded to router 2.

  Through this method, the reordering of packets from the same link can be detected, but not packets from different links. For instance, we cannot detect the reordering of packet 1 and packet a, as the order may depend on the internal scheduling of router 1.

  Depending on the amount of reordering that can be tolerated, we can detect packet reordering attacks by using only the three main steps of the SATS detection engine. In other words, a packet has to be delayed in order

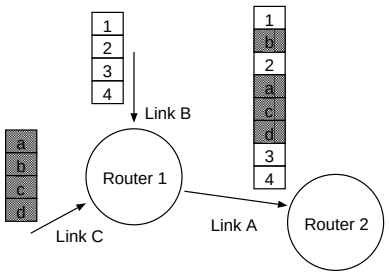


Fig. 14. An example of packet reordering detection. The rectangles at a link represents the time-ordered labels collected from the link.

to reorder a packet. If the delays becomes longer than the trajectory reconstruction timeout (Section II), the trajectory will terminate earlier at the reordering node.

- *Detection of Timing Attacks*: Timing attacks can be detected if routers attach timestamps to the labels they collect. The backend engine responds to an abnormally large differences between the timestamps from two consecutive routers. However, if the routers are not synchronized, choosing a universal threshold for the delay difference may not be easy. This requires further study as well as the issue of clock skew.

  A timing attack can also be detected by the three main steps of the SATS detection engine. If the amount of delay the attack has caused is larger than the trajectory reconstruction timeout, the trajectory will terminate earlier at the delaying node.
- *Identification of Packet Modification and Substitution Attacks*: Once an anomaly is detected, we can use orphan trajectories to identify the packet modification and substitution attacks. As an orphan trajectory starts from the node where the packet is modified, the positive numbers of orphan trajectories suggest that there have been modification or substitution attacks.

IV. Security Analysis

In this section, we address a variety of possible scenarios that could be used to circumvent or confuse SATS. For each scenario, we show how SATS detects the attack and suggest some countermeasures, if necessary.

A. Misreporting, Dropping of Report Packets and Mis-sampling

The basic situation involves a router that simply ceases to report to the measuring node or sends partial or modified reports. Under these circumstances, if the routers continue to forward packets correctly, their behavior will cause a hole to appear in the reconstructed trajectories. We can then easily find the offending router. If the router does not

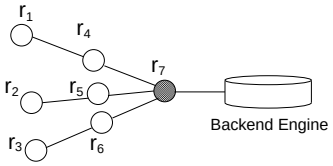


Fig. 15. Dropping of report packets.

forward the reports of other routers, it is also detected as a normal packet drop as reports are transmitted in packets and thus can be sampled. In an extreme case (Fig. 15), a malicious router, r_7, on the way to the backend engine can drop all the report packets from a set of routers, $\{r_1, r_2, r_3\}$. If the backend engine does not receive any report from a router r_i, the routers on the way from r_i to the backend engine are assumed to be suspicious and examined. In this example, $\{r_1, r_4, r_7\}$, $\{r_2, r_5, r_7\}$ and $\{r_3, r_6, r_7\}$ are thus in the suspicious regions. Among the routers in the suspicious regions, the router that overlaps the most, router r_7, is examined first. Thus, if r_7 keeps dropping report packets, it will be the first router to be examined and detected.

If a malicious router does not sample some of the packets in its hash range, or sample wrong packets, the corresponding labels are not reported. It results in an inconsistent trajectory as well.

B. Label Collision Attack

A malicious router, r, can modify a packet p with label l into another packet p' with the same label l, if r can find p' where $h_{label}(p') = h_{label}(p) = l$. Then, the trajectory of p will not end early at r. Thus, the trajectory appears complete, and r can go undetected. In order to prevent this attack, we use a 2nd-preimage resistant hash function (for a given value x, it is computationally infeasible to find a $x' \neq x$ such that $h(x') = h(x)$) to generate labels. Popular checksum algorithms, such as MD-5 or a universal one-way hash function with stronger security and statistical properties [20] can be used. Although we use a cryptographic label hash function h_{label}, the selection hash function, $h_{selection}$, remains the same as that of TS.

C. TTL Modification

Changing the TTL value to a very low number could cause a packet to be dropped by other legitimate downstream routers. This behavior would divert attention away from the real attacker. Such changes would not be caught by C_{prim}, as the label hash function does not operate over variant fields including TTL. This behavior can be prevented

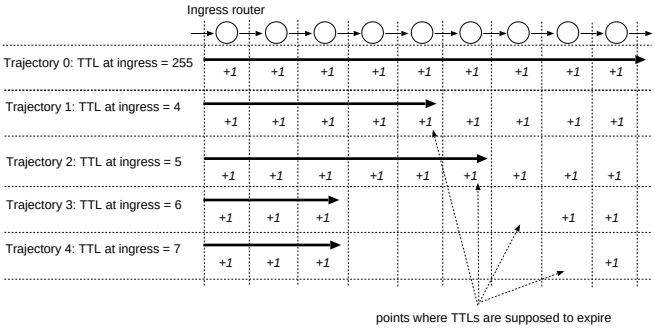


Fig. 16. Solution to reduce the false positives due to TTL adjusted packets. TTL values are reported from the ingress router.

by reporting the TTL value, along with a label, at each sampling node. If the difference between the TTL values from two nodes are more than the number of hops between the nodes, these nodes are suspects.

Some legitimate packets may also be detected as anomalous, especially the packets where TTL is adjusted so that they can be dropped prematurely (e.g. traceroute packets). If the ratio of these packets is high enough to cause false positives, we can compensate for the effect by incrementing the C_{prim}'s for all the nodes after the point where the TTL is supposed to expire. Fig. 16 illustrates such an example. Since trajectories 1 and 2 end where the TTL's are supposed to expire, the additions to C_{prim}'s are the same as trajectory 0 when they are aggregated. On the other hand, trajectory 3 and 4 indicates that they ended earlier.

V. Evaluation

In this section, we evaluate the detection rate, the incremental deployment property, and the full path completion time of SATS. We first describe the simulation method and then present the results.

A. Experiment Setup

We use four topologies published by real networks in our simulation engine. Two are provider networks and two are university campus networks. Table I summarizes them[3].

[3]Maps for ATT, CENIC, Berkeley and CMU can be found at `http://www.ssfnet.org/Exchange/gallery/index.html`, `http://www.cenic.net/operations/maps/`, `http://www.net.berkeley.edu/netinfo/newmaps/` and `http://www.net.cmu.edu/docs/services/ip/core.html`, respectively

Topology	No. of Nodes	No. of Edges	Node Degree	No. Nodes on Shortest Path	Link Metric?
ATT	54	144	1, 1.5, 6	5, 7, 9	No
CENIC	25	54	1, 2, 4	5, 7, 9	Yes
Berkeley	43	112	1, 1, 6.5	4, 5, 6	Yes
CMU	102	250	1, 1, 4.3	4, 6, 6	No

TABLE I

SUMMARY OF TOPOLOGIES USED IN OUR EVALUATION. THE THREE NUMBERS UNDER THE "NODE DEGREE" AND "NO. NODES ON SHORTEST PATH" COLUMNS DENOTE THE 10TH-PERCENTILE, MEDIAN AND 90TH-PERCENTILE VALUES, RESPECTIVELY.

We compute shortest paths between each pair of edge routers in a topology. If link metrics are available for the topology, we use it in the shortest path calculation. Otherwise, we assume all links have equal weights. Our assumption is consistent with most IGP routing protocols that use some variation of link metric when computing shortest paths. As we are concerned with a single autonomous system in this paper, the BGP route selection algorithm, which is heavily dependent on policies not link metrics, does not come into play here.

For the provider networks, we simplify the topology such that each PoP contains only one backbone router and one edge router. In reality there can be two to three backbone routers and up to tens of edge routers connecting customers or peers in a PoP. However, this simplification does not affect our evaluation results, as it does not alter the shortest path results. Edge routers within a PoP do not connect to each other, and they are connected directly to a backbone router. The multiple backbone routers are there for redundancy or are directly connected to the core.

B. Probability of Complete Avoidance

In Single Range Assignment, a malicious router can completely evade detection by targeting packets whose hashes fall outside one universal hash range. In this section, we show that it is very difficult for such an attack to go undetected in Split Range Assignment.

We assume that a malicious router, r_m, randomly selects p_{attack} percentage of hash values out of N_{total} values in the entire hash domain of $h_{selection}$. The number of hash values selected by the malicious router r_m is then $N_{attack} = p_{attack} \times N_{total}$. Let $\mathcal{H}_{attack}(m)$ denote a set of the N_{attack} hash values chosen by r_m. Only the packets whose hashes fall in $\mathcal{H}_{attack}(m)$ are targeted by r_m. Thus, on average, r_m attacks p_{attack} percentage of all the packets.

We define *complete avoidance* to be the event where all the hash values in $\mathcal{H}_{attack}$ are not assigned to any router in a given path. In the case of complete avoidance, the packets being attacked are not sampled at any router, and

h_i: the i-th hash value among N_{total} hash values in the entire hash domain of $h_{selection}$
$\mathcal{H}(i)$: a set of N_{small} hash values in the hash range assigned to the i-th router, r_i
$\mathcal{H}_{enclose}(m) = \{h_i : h_i \in \mathcal{H}(l), h_i \in \mathcal{H}(r), 1 \leq l < m < r \leq T, 1 \leq i \leq N_{total}\}$: a set of hash values assigned to any pair of routers that enclose a malicious router, r_m, on a T-hop path
$N_{enclose}(m)$: the number of hash values in $\mathcal{H}_{enclose}(m) \cup \mathcal{H}(m)$
$\mathcal{H}_{attack}(m)$: a set of hash values chosen by the malicious router r_m.
$N_{attack} = p_{attack} \times N_{total}$: The number of hash values in $\mathcal{H}_{attack}(m)$
P(complete avoidance of r_m) = $p_{avoid}(m)$ = $P(\mathcal{H}_{attack}(m) \cap \mathcal{H}_{enclose}(m) = \phi)$ = $C_{N_{attack}}^{N_{total}-N_{enclose}(m)} / C_{N_{attack}}^{N_{total}-N_{small}}$

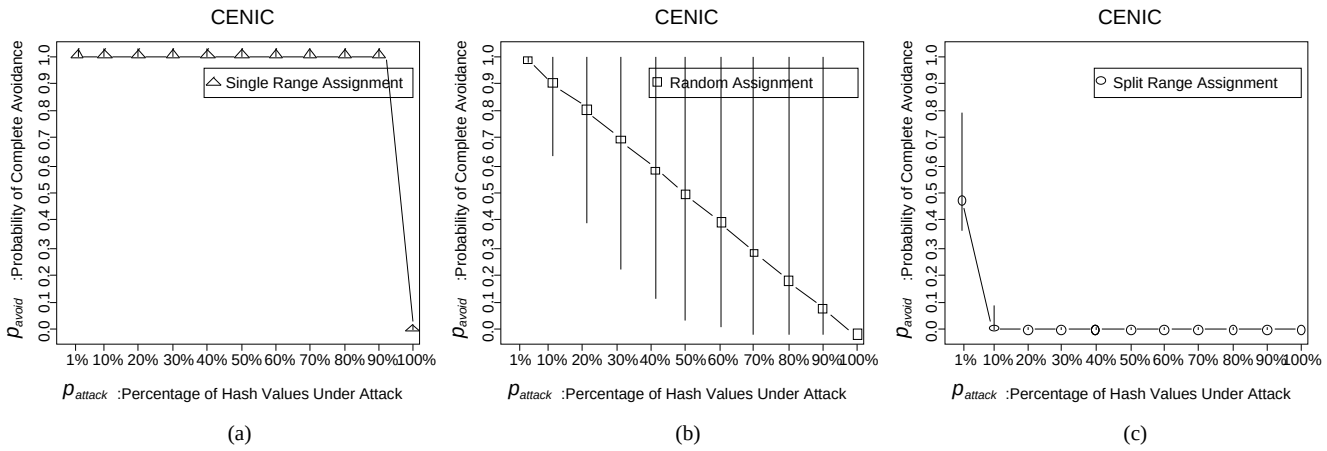
Fig. 17. The calculation of the probability of complete avoidance, p_{avoid}.

the attack cannot be detected. We vary p_{attack} from 1% to 100% in 10% increments and derive the probability of complete avoidance, p_{avoid}. With higher p_{attack}, the malicious router attacks more packets, but the attack is less likely to go undetected leading to lower p_{avoid}.

Fig. 17 shows the derivation of the probability of complete avoidance. We illustrate the derivation through the example in Fig. 3.(b). In this example, we assign 6 hash values to a router as the hash range out of 31 different hash values, $\{h : 1 \leq h \leq 31\}$. Thus, N_{small} = 6, and N_{total} = 31. The 4-th router, r_4, is a malicious router on a 7-hop path. r_4 selects p_{attack} = 20% of the N_{total} hash values ($31 \times 0.2 \approx 6$ hash values) out of $N_{total} - N_{small} = 31 - 6 = 25$ hash values in $\{h : 1 \leq h \leq 31\} - \{25, 15, 28, 14, 12, 8\}$. r_4 does not choose the 6 hash values from its own hash range, $\{25, 15, 28, 14, 12, 8\}$, since these values can also be assigned to other routers. Thus, the number of possible choices is C_6^{31-6}. r_4 then attacks packets whose hashes fall in the 6 chosen hash values, $\mathcal{H}_{attack}$. If any one of the 6 chosen hash values in $\mathcal{H}_{attack}$ is in $\mathcal{H}_{enclose} = \{30, 5, 9, 27, 4, 11, 7, 17, 20\}$, the attack will be detected - the attack on the hash value 30, 5, 9, 27, 4, 11, 7, 17, or 20 will be detected by $\{r_1, r_5\}$, $\{r_1, r_6\}$, $\{r_1, r_7\}$, $\{r_2, r_5\}$, $\{r_2, r_6\}$, $\{r_2, r_7\}$, $\{r_3, r_5\}$, $\{r_3, r_6\}$, or $\{r_3, r_7\}$, respectively. The probability of complete avoidance of r_4, which is the probability that all of the 6 chosen hash values in $\mathcal{H}_{attack}$ are not in $\mathcal{H}_{enclose}$, is $C_6^{31-6-9} / C_6^{31-6} \approx 0.045$.

We assign hash ranges using three different methods, Split Range Assignment (Fig. 4), random assignment and Single Range Assignment. In the random assignment, we assign $N_{small} = N_{router} - 1$ hash values to a router. The hash values are randomly chosen out of $N_{total} = N_{small}/p_{samp}$ hash values in the hash domain of $h_{selection}$. For each node in a topology, we compute the probability of complete avoidance considering all the shortest paths where the node is present. We run 100 simulations with the sampling ratio p_{samp} = 1/1000.

In Fig. 18 we only show the results from the topology of CENIC as the results from the topologies of ATT, Berkeley,



(a) (b) (c)

Fig. 18. Probability of Complete Avoidance. The left column shows results for Single Range Assignment, the middle column for random assignment, and the right column for Split Range Assignment. The median, 10th-percentile and 90th-percentile values of the ratio are shown.

and CMU are very similar. In Single Range Assignment, unless a malicious router attacks all the packets, the malicious router can avoid detection with 100% probability. Random assignment can reduce p_{avoid} but not significantly. In Split Range Assignment, p_{avoid} becomes negligible as p_{attack} approaches 10%. Thus, attacks on more than 10% of packets can hardly circumvent SATS. An attack with p_{attack} less than 1% has almost 50% chance of being undetected. However, in this case, the resulting attack is limited. The attacker has to significantly limit its ability to attack only a small portion of packets. In addition, we can further reduce p_{avoid} by periodically re-launching new hash ranges. Let us assume that the range assignment in each new launch of hash ranges is independent of other launches. A malicious router also changes its target hash values, $\mathcal{H}_{attack}$, independently in each new launch. Then, the probability that the malicious router avoids detection in n consecutive launches of hash ranges is ${p_{avoid}}^n$. After 5 such re-launches ($n = 5$), the probability of complete avoidance decreases to $(50\%)^5 \approx 3\%$ when p_{attack}=1%.

C. Incremental Deployment

In this section, we discuss the size of a suspicious region detected by SATS. We aim to answer the following questions here: What is the size of a suspicious region relative to the diameter of a network? How well does SATS work in different topologies? What is the incremental deployment property of SATS? Is it necessary to have complete deployment before seeing most of the benefits? Is a sophisticated deployment scheme needed to put SATS on strategic routers?

We evaluate two deployment methods: random and degree-based. The degree-based method chooses routers with

the highest number of neighbors first during deployment. It gives priority to securing the more critical routers. We also vary the fraction of routers using SATS, from 0.1 to 1, in 0.1 increments. Eighty different combinations of parameters are based on 10 deployment ratios $\times$ 2 deployment methods $\times$ 4 topologies. 100 simulation runs are made in each combination.

Fig. 19 shows results of the evaluation. The top row is from the random deployment, and the bottom row is from the degree-based. The y-axis plots the suspicious region ratio, computed as the number of nodes in the pinpointed suspicious region divided by the total number of nodes on the shortest path. A node without SATS is considered to be suspicious. The median, 10th-percentile and 90th-percentile values are shown. The x-axis plots the deployment ratio.

All graphs show "knees" in the performance curves: after reaching a certain deployment ratio, from 0.1 to 0.4 depending on the topology and deployment method, the curves flatten out. In other words, we start seeing most of the benefits of SATS before complete deployment on all routers in a network. These "knees" happen because of the hierarchical nature of networks. There are a handful of well-connected routers, which are on most of the shortest paths between pairs of edge routers. Once these routers deploy SATS, the size of a suspicious region can be reduced dramatically.

We see that variability is much lower for the degree-based deployment. In particular, the 90th-percentile values of the suspicious region ratio are much lower for all 4 topologies. The results also show that a practical yet extremely simple method of deploying SATS on routers with the largest degrees first can yield significant improvements over a random method. A sophisticated deployment method may not be necessary. It is more important to secure first the routers with higher degrees of connectivity. We focus on the results of the degree-based deployment below.

When the deployment ratio is zero, the suspicious region ratio is 1 since all of the nodes on the shortest path is considered as suspicious. With half of the nodes in a topology deploying SATS, the median values of the suspicious region ratio range from 0.3 to 0.4, depending on the topology. In other words, we can pinpoint 3 to 4 routers as suspicious routers on a 10-hop path in the presence of malicious routers with the deployment ratio of 0.5.

D. 95% Mean Coverage Count for the Split Range Assignment

In Split Range Assignment, we superpose incomplete trajectories in the same flow to have the full path. This section presents how soon we can recover the path in Split Range Assignment compared to Single Range Assignment. We

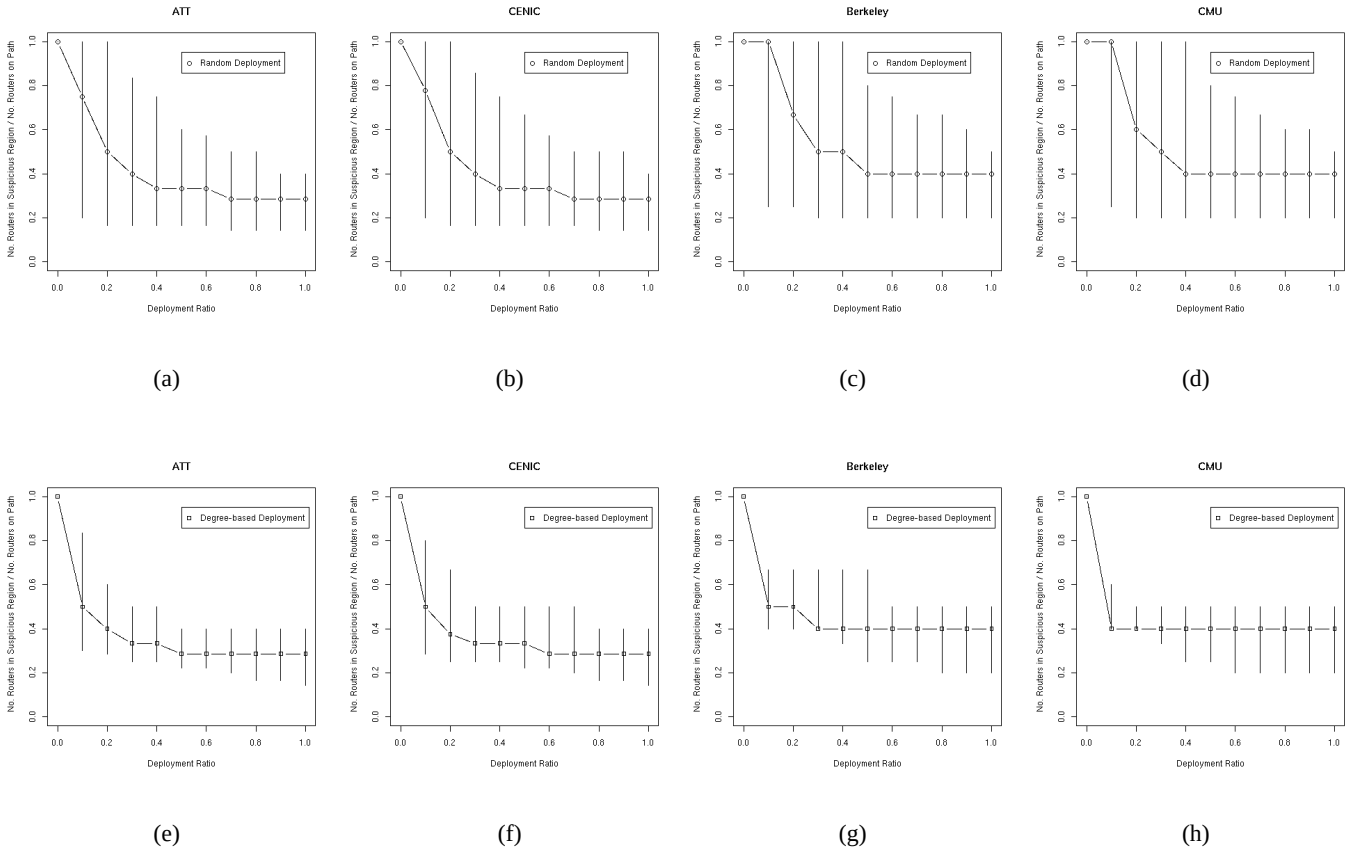


Fig. 19. Suspicious region ratio. The top row shows results from the random deployment, and the bottom row from the degree-based deployment. The median, 10th-percentile and 90th-percentile values of the ratio are shown.

study how Split Range Assignment affects the functions of TS, and whether we have to extend the measurement period set in Single Range Assignment. For instance, if we have to wait a long time to have a complete path for a flow, Split Range Assignment is not suitable for either traffic measurement, or anomaly detection.

We define $min(N)$ to be the minimum number of packets that have to traverse a path before we can complete a path. We define 95% Mean Coverage Count to be $min(N)$ with which we can complete a path with 95% probability.

If we have at least one sampled packet from each hash range, the path becomes complete. We first derive the probability that at least one packet is sampled from each hash range, as a function of n, r and T. n is the number of packets that has traveled a path. r is the number of different hash ranges. T is the number of hops on the path. We then compute the minimum n that leads to 95% probability. We assume that there is no malicious packet drop, as we focus on the ability of SATS (split range assignment) to perform the functions of TS.

Let A denote the number of different hash ranges assigned to a path, and S denote the number of hash ranges where

we have at least one sampled packet. The 95% Mean Coverage Count $min(N)$ is obtained when $P(S = A \mid N = min(N)) \geq 0.95$ is satisfied. If we start a *random* assignment of r different hash ranges, the number of hash ranges assigned on the path varies from 1 to r. Thus, we first derive the conditional probability $P(S = a \mid A = a, N = n)$ assuming that the number of assigned ranges is a constant a. We then derive $P(A = k \mid N = n)$ for $k = 1$ to r to get

$$\begin{aligned} P(S = A \mid N = n) &= \textstyle\sum_{k=1}^{r} P(S = A \mid A = k, N = n) \\ &\quad \times P(A = k \mid N = n) \\ &= \textstyle\sum_{k=1}^{r} P(S = k \mid A = k, N = n) \\ &\quad \times P(A = k \mid N = n) \end{aligned} \tag{1}$$

Let $P(S = a \mid A = a, N = n)$ be the probability that a path is complete after N packets have traversed the path. We model SATS as a Markov Chain in which a state represents the number of ranges where at least one packet has been sampled. We assume hash ranges are distinct, and are randomly assigned. If $A = a$, the transition matrix, R_a, of the Markov Chain is given as follows.

$$R_a(i,j) = \begin{cases} 1 - \dfrac{(a-i)}{(\frac{1}{p_{samp}})}, & j = i \\ \dfrac{(a-i)}{(\frac{1}{p_{samp}})}, & j = i+1 \\ 0, & otherwise \\ & (0 \leq i \leq a,\ 0 \leq j \leq a) \end{cases} \tag{2}$$

State $R_a(i,j)$ is the transition probability from the state i to the state j after a packet cross the path. There is initially no sampled packet. So, we start from state 0. We then derive the probability that the state is a after n packet has crossed the path. Thus, $P(S = a \mid A = a, N = n) = R_a^n(0,a)$. $R_a^n(0,a)$ is the element of R_a^n located at row 0, column a.

$P(A = a \mid N = n)$ is the probability that a hash ranges are finally assigned on a T-hop path when we initially start range assignment with r hash ranges. We model as another Markov Chain, where a state represents the number of distinct ranges assigned. Then the transition matrix M_r is given as follows.

$$M_r(i,j) = \begin{cases} 1, & j = 1 \; and \; i = 0 \\ \frac{i}{r}, & j = i \; and \; i > 0 \\ \frac{(r-i)}{r}, & j = i+1 \; and \; i > 0 \\ 0, & otherwise \\ & (0 \le i \le r, \; 0 \le j \le r) \end{cases} \tag{3}$$

$M_r(i,j)$ represents the probability of transition from state i to state j after a hash range is assigned to a link. Then $P(A = a \mid N = n) = M_r^T(0,a)$. $M_r^T(0,a)$ is the element of M_r^T at the intersection of row 0, column a. Therefore, according to (1),

$$\begin{aligned} P(S = A \mid N = n) &= \textstyle\sum_{k=1}^{r} P(S = k \mid A = k, N = n) \\ &\quad \times P(A = k \mid N = n) \\ &= \textstyle\sum_{k=1}^{r} R_k^n(0,k) \cdot M_r^T(0,k) \end{aligned} \tag{4}$$

We plot $min(N)$ where $P(S = A \mid N = n)$ reaches 95% in Fig. 20. The sampling ratio is 0.05. The maximum number of ranges, $max(r)$, is then $\frac{1}{p_{samp}} = 20$. As r increases, $min(N)$ does not grow more than twice of $min(N)$ with a single hash range ($r = 1$). The more hash ranges r we use, the more holes we have on each trajectory. But the more trajectories are sampled from the same number of packets with sampling ratio $r \cdot p_{samp}$. Thus, the use of multiple hash ranges only slightly affects the path completion time. The growth of T has a similar impact on $min(N)$ to the growth of r, since more hash ranges will be assigned from r hash ranges as T becomes larger.

VI. Conclusion

In this paper, we present SATS, an effective method that detects malicious routers on the data plane. SATS utilizes the traffic measurement infrastructure, TS, which is actively being developed in IETF for standardization. Packets are sampled based on TS mechanisms, and Split Range Assignment ensures that the sampled packets reflect the behavior of all the packets even in the presence of malicious routers. SATS pinpoints a suspicious router region and detects a single and consecutive malicious routers. Split Range Assignment varies hash ranges in order to maintain the integrity of sampling packet process. The probability that a malicious router avoids detection in Split Range Assignment is less

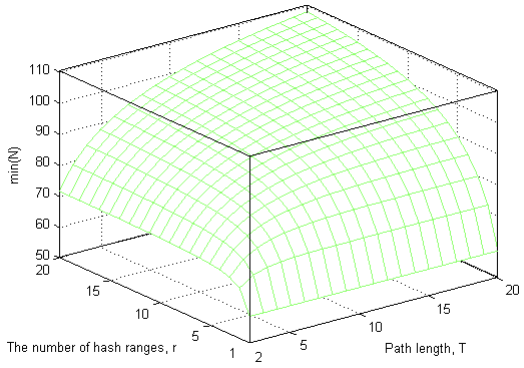


Fig. 20. Mesh plot of 95% coverage count of N as a function of path length T and the number of hash ranges r. The sampling ratio p_{samp} is 5%.

than 5%, and with periodic re-launch of new hash ranges, the probability becomes even lower. In addition, SATS can be deployed incrementally. We see most of the benefits of SATS when 10 to 40% of the routers in a network deploy SATS. Although SATS is initially designed for a single administrative domain, it can also be extended to multiple non-cooperative domains by sharing common hash values in border routers.

REFERENCES

[1] B. Greene and K. Houle, "Isp security – real work techniques ii," NANOG 26 presentation, October 2002, http://www.nanog.org/mtg-0210/ispsecure.html.

[2] R. Bush, R. Thomas, N. Ziring, and G. Jones, "Simple router security, what every isp router engineer should know and practice," NANOG 29 presentation, October 2003, http://www.nanog.org/mtg-0310/routersec.html.

[3] B. Schneier, "Cisco Harasses Security Researcher," July 2005, http://www.schneier.com/blog/archives/2005/07/cisco_harasses.html.

[4] O. Nordstrm and C. Dovrolis, "Beware of bgp attacks," *ACM Sigcomm CCR*, April 2004.

[5] "Router security configuration guide," System and Network Attack Center, National Security Agency, December 2003, Available at http://www.nsa.gov/snac/routers/cisco_scg-1.1b.pdf.

[6] S. Kent, C. Lynn, and K. Seo, "Secure border gateway protocol (secure-bgp)," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, April 2000.

[7] "IETF Routing Protocol Security (rpsec) Working Group," http://www.ietf.org/html.charters/rpsec-charter.html.

[8] V. Padmanabhan and D. Simon, "Secure traceroute to detect faulty or malicious routing," *Sigcomm Computer Communications Review*, vol. 33, no. 1, pp. 77–82, Jan 2003.

[9] I. Avramopoulos and J. Rexford, "Stealth Probing: Securing IP Routing through Data-Plane Security," Princeton University Department of Computer Science, Tech. Rep. TR-730-05, June 2005.

[10] J. Huges, T. Aura, and M. Bishop, “Using Conservation of Flow as a Security Mechanism in Network Protocols,” in *IEEE Symposium on Security and Privacy*, May 2000.

[11] A. T. Mizrak, Y. Cheng, K. Marzullo, and S. Savage, “Fatih: Detecting and isolating malicious routers,” in *Dependable Systems and Networks*, Yokohama, Japan, June 2005.

[12] N. Duffield and M. Grossglauser, “Trajectory sampling for direct traffic observation,” *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 280–292, June 2001.

[13] “Cisco Netflow,” http://www.cisco.com/warp/public/732/Tech/netflow.

[14] C. Estan, K. Keys, D. Moore, and G. Varghese, “Building a Better NetFlow,” in *SIGCOMM*, August 2004.

[15] N. Duffield, A. Gerber, and M. Grossglauser, “Trajectory engine: A backend for trajectory sampling,” in *IEEE Network Operations and Management Symposium 2002*, Florence, Italy, April 2002.

[16] N. Duffield and M. Grossglauser, “Trajectory sampling with unreliable reporting,” in *IEEE INFOCOM*, Hong Kong, March 2004.

[17] “IETF IP Flow Information Export (ipfix) Working Group,” http://www.ietf.org/html.charters/ipfix-charter.html.

[18] “IETF Packet Sampling (psamp) Working Group,” http://www.ietf.org/html.charters/psamp-charter.html.

[19] N. G. D. (Ed.), A. Greenberg, M. Grossglauser., J. Rexford, D. Chiou, B. Claise, P. Marimuthu, and G. Sadasivan, “A framework for packet selection and reporting,” July 2005, internet Draft. Available at http://www.ietf.org/internet-drafts/draft-ietf-psamp-framework-10.txt.

[20] M. Naor and M. Yung, “Universal one-way hash functions and their cryptographic applications,” in *ACM Symposium on Theory of Computing*, Seattle, WA, May 1989.