Packet Forwarding Misbehavior Detection in Next-Generation Networks

Vikram Desai, Sriram Natarajan and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
{vdesai,snataraj,wolf}@ecs.umass.edu

***Abstract*—The next-generation Internet promises to provide a fundamental shift in the underlying architecture to support dynamic deployment of network protocols. With the introduction of programmability and dynamic protocol deployment in routers, potential vulnerabilities and attacks are expected to increase. In this paper, we consider the problem of detecting packet forwarding misbehavior in routers. Specifically, we focus on an attack scenario, where a router selectively drops packets destined for another node. Detecting such an attack is challenging since it requires differentiating malicious packet drops from congestion-based packet losses. We propose a controller-based misbehavior detection technique that effectively detects malicious routers using a hash-based delay sampling and verification. We provide a performance analysis of the detection accuracy and quantify the performance overhead of our system. Our results show that our technique provides accurate detection with low sampling rates.**

I. Introduction

The next-generation Internet promises to provide a fundamental shift in the underlying architecture to support the increase in demand of deploying dynamic, heterogenous applications. Virtualized network infrastructures allow multiple logical networks to coexist on a shared physical infrastructure platform, as discussed in [1], [2]. The idea of introducing programmability in the data-path of routers facilitates the dynamic deployment of new protocol stacks and services [3]–[5]. The customization of routers using programmability and shared network virtualization platform introduces increased vulnerabilities and attacks on the router platform, as shown in [6].

A key problem in networks, where traffic is forwarded by multiple routers that belong to different administrative entities, is that there is no easy way of detecting misbehaving network nodes. Malicious routers may drop packets instead of forwarding (i.e., black hole attack) or drop some fraction of packets (i.e., gray hole attack). While an end-system may detect that certain packets are not being received, it is difficult to infer which node dropped these packets. Even more difficult is to determine if a packet drop is due to malicious behavior or due to network conditions (e.g., congestion, bit errors in link layer, etc.). To address this problem, techniques for verifying the correct behavior of routers have been developed. In related work, initial ideas for explicit verification of forwarding actions (i.e., proof of forwarding) have been developed. A concern in these systems is the overhead introduced by this verification, since most techniques involve the end systems in the detection process.

In this paper, we propose a secure controller based forwarding misbehavior detection system that uses a hash-based delay sampling algorithm and a verification technique to detect the malicious packet drops introduced by routers. At a given time, the controller randomly chooses set of three nodes (a router node to monitor and its corresponding uplink and downlink neighbor nodes) in the forwarding path and gathers the forwarding information (sample aggregate) from the monitored node and the evidence information (i.e., the set of packets forwarded by the uplink node and the actual packets received by the downlink node) from the neighbor nodes. This technique localizes the identification of misbehaving modes, thereby reducing the overhead within the network since only a sampled subset of forwarding actions is required. The specific contributions of our paper are: 1) formulation of the malicious packet forwarding router problem, 2) design a forwarding misbehavior detection system to accurately determine the malicious router, 3) presentation of a performance analysis of the proposed technique.

The reminder of the paper is organized as follows. Section II discusses the related work. Section III introduces the design of the proposed forwarding misbehavior detection system. Section IV discusses the security model explaining the requirements and attacker capabilities. Section V proposes our defense mechanism, explaining the sampling algorithm and the verification technique. Section VI discusses the performance analysis of the proposed system and Section VII summarizes and concludes the paper.

II. Related Work

The problem of identifying the compromised routers on a given forwarding path can be determined by either developing a protocol based adversary identification, as shown in [7], [8] or using a distributed monitoring approach that involves traffic validation mechanisms to evaluate the performance of each router. Reference [9] presents an effective fault tolerant forwarding technique that is effective with Byzantine failures (e.g., forging or modifying protocol messages). Reference [10] shows a detection scheme by considering the data and ack packet characteristics to monitor the selected route from source to destination. The technique monitors the end to end connectivity but cannot identify the compromised routers in the path.

Reference [11] presents a protocol that provides a combination of source routing, hop based authentication, and end to end reliability mechanism. However, the scheme introduces significant storage and communication overhead. Reference [12] proposes a distributed detection system that uses traffic validation schemes between the source and the intermediate routers. The detection technique suggested in [13] can at best detect a faulty path, but cannot help come to a consensus as to which router along the faulty path is presenting malicious behavior. Both techniques involve significant performance overhead and provide low detection accuracy in determining if a monitored node is malicious. Current distributed monitoring approaches involve intermediate routers or end systems in the forwarding path to detect the misbehaving routers. This introduces significant communication and storage overhead to validate the router performance.

III. Packet Forwarding Misbehavior Detection

In this paper, we propose a controller-based monitoring technique that gathers forwarding behavior from neighboring nodes that is efficient in detection accuracy and introduces lower performance overhead. Currently, the proposed technique focuses on detecting malicious packet drops from a compromised router. Before discussing the design of our system, we enumerate the set of assumptions that are considered in our design.

A. Assumptions

Modern routers have sufficient capability to process packets and provide aggregate data describing details on the traffic they are forwarding. We presume that the routers have computational power to send the traffic aggregate to neighboring nodes as well as receive messages from them. The proposed system is designed to detect a compromised router in the core network and assumes the access routers to never be faulty in terms of the traffic originating or ending at the router. We consider the following assumptions to monitor the compromised node using the controller:

- It is possible to establish a secure connection between the controller and each individual router nodes (e.g., SSL).
- All routers maintain reasonably synchronized clocks (e.g., the granularity achievable by NTP [14]).
- Individual routers are connected by a duplex link and forward packets by looking up forwarding tables in a multi-hop fashion.
- There is no collusion among the compromised routers in the network.
- The path a packet takes is known a priori (it has been shown that routers can predict the path for a packet once the network has become stable).

B. Design

The technique of using witness-based forwarding misbehavior detection in wireless networks was shown in [15]. Unlike the witness-based model, which chooses the witness nodes from the set of observing neighbor nodes, we consider a secure controller node (for a given autonomous system) that dynamically picks and collects sample aggregate from the forwarding node and evidence aggregates from its neighbor nodes.

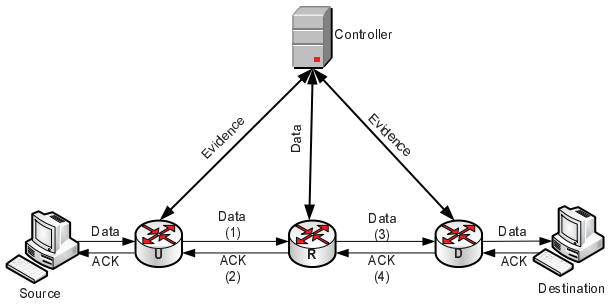

Fig. 1. Forwarding Misbehavior Detection System

The design of the forwarding misbehavior detection system is shown in Figure 1. The system consists of a controller node (C, that does not lie on the forwarding path of the router), the monitored router (R), and its corresponding uplink (U) and downlink (D) routers and the end systems. The detection process involves the following steps: Consider a forwarding path with intermediate routers $U - R - D$ as shown in Figure 1. At time t, the controller node C monitors the forwarding behavior of router R (We assume that nodes U and D are secure at that instance). To perform an effective monitoring, router R send the sampled aggregate statistics to C and routers U and D together send the sampled evidence statistics to C. The controller then performs a verification mechanism on the received packets to determine the performance of the monitored node R. A detailed description of the sampling algorithm and the verification process is discussed in Section V.

IV. Security Model

In this section, we discuss our security model, describing the security requirements and attacker capabilities that can introduce compromised packet forwarding behavior in routers.

A. Security Requirements

The following set of security requirements ensure the secure processing of network traffic by routers:

- Ensure correct packet forwarding behavior of all routers in the network.
- Identify routers that introduce malicious packet forwarding behavior.
- Infer and discard malicious traffic originating from the compromised router.

B. Attacker Capabilities

The following attacker capabilities define the potential attack scenarios that can be launched from the compromised routers:

- The attacker can selectively drop legitimate network traffic, which introduces malicious packet loss behavior by exploiting the congestion control mechanism.

- The attacker can send arbitrary network traffic (data and control packets) from the compromised router.
- The attacker can modify the data packet to introduce anomalous forwarding and routing behavior.
- The attacker can physically tamper or remotely access the router to extract secure information from the device.

V. Defense Mechanism

In this section, we discuss the design challenges and functionalities of our proposed controller based forwarding misbehavior detection system.

A. Challenges

The fundamental challenge of our misbehavior detection technique is to provide an accurate detection technique with low false positive (identifying congestion based packet loss as malicious packet drop) and false negative (incorrectly disregarding a malicious packet loss as congestion based packet loss) detection rates. Another important challenge of the proposed mechanism is to provide a technique that is effective with respect to the communication overhead required to detect a malicious forwarding behavior (packet loss) and the storage overhead involved between the controller and the router nodes.

To address the above challenges, the proposed mechanism involves 1) a robust sampling technique that introduces lower performance overhead and 2) a verification technique that is effective in detecting the malicious packet processing behavior.

B. Sampling

In this section, we describe our hash-based delay sampling technique, explaining the functionalities performed by the monitored node and the evidence nodes to provide the required sample aggregate to the controller node. The concept of delay sampling for verifiable network measurements was proposed in [16]. Delay sampling requires each router to maintain network state on all observed packets for a fixed interval of time. The sampling is then performed on the stored set of packets. The advantage of this technique is to prevent a compromised router from giving preferences to the sampled set of packets, as it could forward the sampled set of packets correctly and drop the packets that would not be added to the sample subset.

We modify the sampling algorithm proposed in [16] to support the monitoring of multiple paths (maintaining individual buffers for each path) at a given time, as shown in Figure 2. For each path $path_i$ the buffer maintains n tuples $(T_1, T_2, ..., T_n)$. The advantage of such a scheme is to improve the detection accuracy and also identify the packets belonging to the compromised path. The fundamental requirement of the sampling algorithm is to decide on when to initiate the sampling process in the routers. To avoid sending excessive control information to initiate the sampling process, the source sends a data packet whose initiator value is below a predefined initiator threshold. The initiator packet initiates the sampling process at the router R and the sample aggregate (set of tuples) are sent to C. Similarly, upon receiving the initiator packet, U and D sample the evidence packets and send the sample aggregate (evidence tuples) to the controller node for verification.

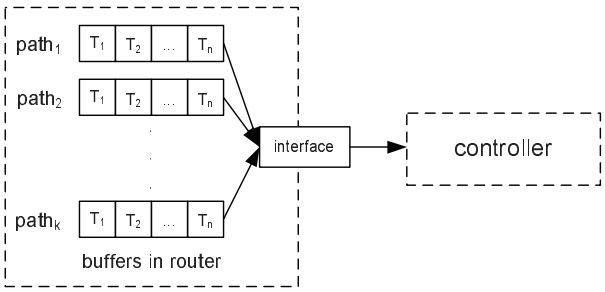

Fig. 2. Buffers in Routers

TABLE I
Sampling Details

Variables	Definition
p	new packet
T_p	packet tuple
$PathID_i(p)$	packet's path
$PacketID_i(p)$	hash function
$InitiatorID_i(p)$	hash function
η	initiator threshold
σ	sampling threshold
B_i	circular buffer for a path i
S_i	sample aggregate for a path i

The set of sample variables and the associated definitions used in the sampling algorithm are shown in Table I. Each node associates a packet with a $PathID$ that defines the packet's path and computes a set of hash functions ($PacketID$, $InitiatorID$). Each hash function value is generated using a cryptographically secure hash function (e.g., SHA-1 [17]). In addition, the node maintains k circular buffers (B_i) for the recently seen k unique paths and k sample aggregate (S_i), which are sent to the controller node. The process of maintaining k circular buffers at a given instance of time improves the detection accuracy compared to observing a single path. The creation and deletion of the buffers are based on the k recent paths observed in the router. For simplicity, we show the packet sampling process for a single path.

Algorithm 1 shows the sampling algorithm when a new packet is received. First the node computes (for path i) the $PathID_i(p)$, $PacketID_i(p)$, and $InitiatorID_i(p)$ for the received packet. The algorithm then computes a tuple T_p ($PathID_i(p)$, $PacketID_i(p)$) (line 2) for each of the n recently observed packets in the path. The sampling is initiated when an initiator packet is received for the corresponding path. An initiator packet is identified by calculating the hash ($InitiatorID$) of the packet. If the hash value is below the initiator threshold (η) (line 3), then the observed packet is treated as an initiator packet, else, the calculated tuple is added to the corresponding buffer B_i (line 13).

Next, the algorithm determines the set of tuples in the buffer B_i whose hash value (line 5) is below the sample threshold (σ) value. The set of tuples that satisfy this condition is

Algorithm 1 Sample $<p>$

Require: $B_i \leftarrow 0$, $S_i \leftarrow 0$

1: **for** $i = 1$ **to** k **do**
2: $T_p = (PathID_i(p), PacketID_i(p))$
3: **if** $InitiatorID_i(p) \leq \eta_i$ **then**
4: **for** all T in Bi **do**
5: **if** Hash(T_p,T) $\leq \sigma$ **then**
6: $S_i \leftarrow T$ $\{Sample\ the\ new\ packet\}$
7: **else**
8: $Clear\ T\ from\ B_i$
9: **end if**
10: **end for**
11: $S_i \leftarrow T_p$ $\{Sample\ the\ initiator\ packet\}$
12: **else**
13: $B_i \leftarrow T_p$
14: **end if**
15: **end for**

sampled out and added to the sample aggregate (S_i) (line 6). the remaining tuples in B_i are removed from the buffer (line 8). The initiator packet's tuple is also added to the sample aggregate buffer (line 10). Once the sampling is completed, the total sample aggregate S_i is sent to the controller node for verification. A similar sampling method is initiated at the corresponding uplink and downlink routers to sample and send the evidence tuples to the controller node for verification. The initiator packet ensures that the same set of packets are sampled at each node.

The probability that the new packet is an initiator packet is given by (η/M), where η represents the initiator threshold and M is the maximum value generated by the hash function ($InitiatorID_i(p)$). If the new packet (p) is not an initiator packet and if the buffer is not full, then the probability of seeing an initiator packet after p is given by the cumulative distribution function (CDF) as:

$$CDF = 1 - (1 - \frac{\eta}{M})^n \tag{1}$$

The probability that a given packet's tuple in the buffer B_i is sampled is given by σ/S, where σ represents the sample threshold and S is the maximum value generated by the $PacketID_i$(p). Hence the probability that the packet's tuple is included in the sample aggregate is:

$$P_{T_p} = (1 - (1 - \frac{\eta}{M})^n) \times \frac{\sigma}{S} \tag{2}$$

C. Verification

The verification is performed at the controller node C when it receives the sample aggregate from the router (R) and the evidence aggregates from the uplink (U) and downlink (D) nodes. The hash-based delay sampling ensures that the three nodes, U, R, and D, sample the same set of packets. The verification process performs a comparison on the aggregates received as follows: 1) For each tuple received, the controller C first XOR's the tuple bits of R and U, 2) C then XOR's the tuple bits of R and D, 3) the hash values of the output of step 1 and step 2 are computed 4) the controller then compares the hash values. If the hash values match, then the router's current state is determined to be not compromised. If the hash values do not match, then the controller determines that the router is compromised. If the percentage of loss observed for a given set of packets (window size) is higher than the expected congestion based loss, then a malicious behavior is detected.

The performance of the verification process can be improved by comparing the samples using tuple aggregates rather than checking each tuple in the sample aggregate, as shown in [18]. We plan to incorporate this technique in our future work. The analysis on the accuracy of our current detection technique is discussed in the next section.

TABLE II
FALSE POSITIVE/NEGATIVE ANALYSIS

classification	malicious (actual)	benign(actual)
loss as malicious	true positive	false positive
loss as benign	false negative	true negative

TABLE III
DETECTION ACCURACY PROBABILITY DEFINITIONS

probabilities	events
P[A]	actual malicious loss
P[B]	actual benign loss
P[X]	classified malicious loss
P[Y]	classified benign loss
P[X|A]	true positive
P[X|B]	false positive
P[Y|A]	false negative
P[X|B]	false positive

VI. Analysis

In this section, we discuss the performance analysis of our proposed technique. In our analysis, we actively monitor the network traffic traversing a router that exhibits a Byzantine failure scenario, switching between benign and malicious states. To model this behavior, we consider a two-state Markov model to represent the benign and malicious states of the router. Also, to model the malicious router state, we consider an active sample collection rather than injecting additive losses on network traces. The problem of identifying a malicious router on a given forwarding path requires a technique that introduces 1) efficient detection accuracy in determining the functioning of the router, and 2) lower performance overhead. We provide an analysis for the above two performance metrics.

A. Detection Accuracy

The detection accuracy metric is determined by evaluating the false positive rates. In our analysis, we assume the packet loss events to be independent. Let event A represent the actual malicious loss and event B represent the actual benign (congestion) loss Let event X denote the classified malicious loss and event Y denote the classified benign (congestion)

loss, as seen by our detection mechanism. The false positive/negative rates are determined by applying Bayes' theorem and the scenarios are shown in Table II. Table III shows the set of probabilities that is required to determine the detection accuracy of the system. The false positive is given by the probability of classifying an actual benign loss packet as a malicious packet and is represented as:

$$P[B|X] = \frac{P[X|B] \times P[B]}{P[X]} \tag{3}$$

Similarly, the false negative is given by the probability of classifying a malicious packet loss as a benign (congestion) loss and is represented as:

$$P[A|Y] = \frac{P[Y|A] \times P[A]}{P[Y]} \tag{4}$$

For time t, with window size of w packets, we randomly introduce varying packet loss percentage in the malicious state and in the benign state (e.g., the monitored router would be in the benign state for $x\%$ of the time and in a malicious state for the remaining $y\%$ in the monitored time period). When the $InitiatorID$ of the received packet is less than η, we start sampling the tuples from the corresponding buffer. The set of tuples that satisfy the sampling threshold (σ) condition are sampled out and added to the sample aggregate. The sample aggregate is then sent to the controller for verification. The verification process then evaluates the tuples and classifies the packet loss behavior to be either in good or bad state.

We now discuss the method of determining the false positive and true positives rates in our verification mechanism. To evaluate the performance of the proposed system, we analyze a window of about 1000 packets for a specific time period. In the above window, we randomly inject packet losses. The injected packet losses simulate having benign (congestion) losses in the good state and malicious losses in the bad state. For the window of 1000 packets, we sample out packets from a buffer size of $w = 20$. We vary the sampling rates ranging from 10% to 90% (set of sample packets whose hash value is less than the sampling threshold σ), with step size of 10%. The detection accuracy is calculated by comparing the actual injected loss behavior with our sample output aggregate for various sampling thresholds.

Figure 3 shows the ROC curve explaining the detection accuracy of our system for varying sampling rates (S= 40%, 60%, and 80%). To evaluate the performance of our proposed controller-based detection system, we compare our scheme with HSER [11], which employs a detection technique using intermediate nodes in the forwarding path, and Secure-traceroute [12], which utilizes the end nodes in detection of packet loss. To determine the optimal sampling rate and the permissible threshold value of losses, we conducted multiple analysis by varying the allowable congestion based loss rate and threshold rate for malicious behavior. Our proposed system can accurately determine the malicious behavior at an attack threshold of 20%. The accuracy for each sampling rate is shown in Table IV and is discussed below.

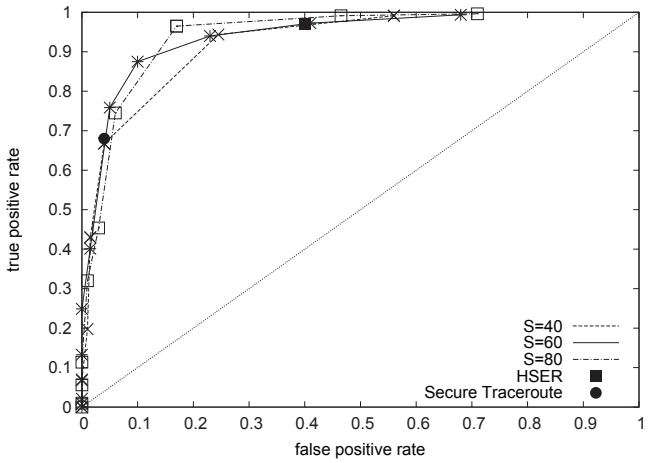

Fig. 3. ROC Curve

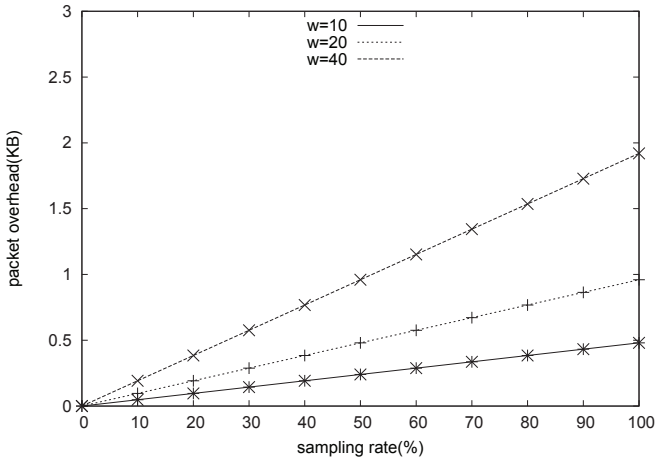

Fig. 4. Sample Threshold vs Packet Overhead (KB)

B. Performance Overhead

The total packet overhead required in our system is determined by the summation of the size of the sample aggregate sent to the controller from the monitored router and the size of evidence aggregates sent from the uplink and downlink nodes in the forwarding path. The size of each sample aggregate is given by the product of the number of tuples and the size of each tuple. The number of tuples is dependent on the sampling rate. The tuple size is given by the summation of the size of the $PathID_i(p)$ of the packet and the $PacketID_i(p)$ of the packet.

The tuple size consists of a hash value of the 5-tuple fields (i.e., IP source, IP destination, source port, destination port, and IP protocol) (size = 104 bits) of a packet and the path id (size = 20 bits), which is approximately about 16 bytes. We consider the overhead involved in transmitting the tuple aggregates at various sampling threshold rates. Since all the nodes (U, R, and D) in the setup sample at the same rate, the overhead involved in transmitting the tuples to the controller node is three times the size of the sample aggregate

TABLE IV
PERFORMANCE COMPARISON ANALYSIS

technique	overhead (KB)	accuracy
40% sampling	0.38	0.90
60% sampling	0.57	0.91
80% sampling	0.76	0.92
HSER	2.14	0.78
Secure-traceroute	9.36	0.82

of the monitored router (i.e., the summation of the size of the monitored node, the uplink and the downlink node). The packet overhead performance of our system for different buffer sizes (w = 10, 20, and 40) is shown in Figure 4. The graph shows the overhead for different sampling rates in the x-axis and packet overhead (in KB) in the y-axis.

Table IV shows the performance comparison of the proposed system with the HSER [11] and Secure-traceroute [12] techniques. Clearly, the accuracy of detection and the packet overhead for one detection cycle of our proposed system performs better when compared with the above techniques. The above set of analysis show that the proposed detection system is effective in detection accuracy and introduces lower performance overhead.

VII. Summary and Conclusion

In this paper, we have proposed a packet forwarding misbehavior detection system that monitors the forwarding behavior of routers on the data path. Unlike previous detection systems that rely on detection at the end systems or identifying the faulty paths, we provide a controller-based detection technique that utilizes the sample aggregate from the monitored node, its corresponding uplink and downlink nodes. The performance analysis of our proposed mechanism shows the introduction of lower packet overhead and effective detection accuracy with lower false positive rates. Currently, the verification process is performed on tuple basis. In the future, we plan to modify the verification by comparing tuple aggregates. Such modifications can optimize the detection time observed at the controller.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Nos. 0952524 and 1115999.

References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.

[2] J. S. Turner and D. E. Taylor, "Diversifying the Internet," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, vol. 2, Saint Louis, MO, Nov. 2005.

[3] A. Feldmann, "Internet clean-slate design: what and why?" *SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 59–64, Jul. 2007.

[4] T. Wolf, "In-network services for customization in next-generation networks," *IEEE Network*, vol. 24, no. 4, pp. 6–12, Jul. 2010.

[5] R. Dutta, G. N. Rouskas, I. Baldine, A. Bragg, and D. Stevenson, "The SILO architecture for services integration, control, and optimization for the future Internet," in *Proc. of IEEE International Conference on Communications (ICC)*, Glasgow, Scotland, Jun. 2007, pp. 1899–1904.

[6] D. Chasaki, Q. Wu, and T. Wolf, "Attacks on network infrastructure," in *Proc. of Twentieth IEEE International Conference on Computer Communications and Networks (ICCCN)*, Maui, HI, Aug. 2011.

[7] X. Zhang, A. Jain, and A. Perrig, "Packet-dropping adversary identification for data plane security," in *Proceedings of the 2008 ACM CoNEXT Conference*, ser. CoNEXT '08. New York, NY, USA: ACM, 2008, pp. 24:1–24:12. [Online]. Available: http://doi.acm.org/10.1145/1544012.1544036

[8] S. Alexander, B. DeCleene, J. Rogers, and P. Sholander, "Requirements and architectures for intrinsically assurable mobile ad hoc networks," in *Proc. of the 2008 IEEE Conference on Military Communications (MILCOM)*, San Diego, CA, Nov. 2008.

[9] R. Perlman, "Network layer protocols with byzantine robustness," Ph.D. dissertation, Massachusetts Institute of Technology, 1989.

[10] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz, "Listen and whisper: security mechanisms for bgp," in *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251175.1251185

[11] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy, "Highly secure and efficient routing," in *IN PROC. IEEE INFOCOM 2004, HONG KONG*, 2004.

[12] V. N. Padmanabhan and D. R. Simon, "Secure traceroute to detect faulty or malicious routing," *SIGCOMM Computer Communications Review*, vol. 33, pp. 77–82, 2002.

[13] A. T. Mizrak, Y.-C. Cheng, K. Marzullo, and S. Savage, "Detecting and isolating malicious routers," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 3, pp. 230–244, Jul-Sep 2006.

[14] O. Gurewitz, I. Cidon, and M. Sidi, "Network classless time protocol based on clock offset optimization," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 876–888, August 2006. [Online]. Available: http://dx.doi.org/10.1109/TNET.2006.880181

[15] S. Yang, S. Vasudevan, and J. Kurose, "Witness-based detection of forwarding misbehaviors in wireless networks," in *Wireless Mesh Networks (WIMESH 2010), 2010 Fifth IEEE Workshop on*, june 2010, pp. 1 –6.

[16] K. Argyraki, P. Maniatis, and A. Singla, "Verifiable network-performance measurements," in *Proceedings of the 6th International COnference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 1:1–1:12. [Online]. Available: http://doi.acm.org/10.1145/1921168.1921170

[17] D. E. Eastlake and P. E. Jones, "US secure hash algorithm 1 (SHA1)," Network Working Group, RFC 3174, Sep. 2001.

[18] R. R. Kompella, K. Levchenko, A. C. Snoeren, and G. Varghese, "Every microsecond counts: tracking fine-grain latencies with a lossy difference aggregator," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 255–266. [Online]. Available: http://doi.acm.org/10.1145/1592568.1592599