2012

Techniques for Detection of Malicious Packet Drops in Networks

Vikram R. Desai
University of Massachusetts Amherst

TECHNIQUES FOR DETECTION OF MALICIOUS PACKET DROPS IN NETWORKS

A Thesis Presented

by

VIKRAM RAGHAVENDRA DESAI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

SEPTEMBER 2012

Electrical and Computer Engineering

TECHNIQUES FOR DETECTION OF MALICIOUS PACKET DROPS IN NETWORKS

A Thesis Presented

by

VIKRAM RAGHAVENDRA DESAI

Approved as to style and content by:

Tilman Wolf, Chair

Weibo Gong, Member

Michael Zink, Member

C.V.Hollot, Department Head
Electrical and Computer Engineering

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Tilman Wolf, for guiding me in shaping my research problem and then for all his encouragement and support during my time in UMass, Amherst. Also, my deep appreciation to my fellow researchers and collaborators, Sriram Natarajan, Shashank Shanbhag for their hard work and support.

ABSTRACT

TECHNIQUES FOR DETECTION OF MALICIOUS PACKET DROPS IN NETWORKS

SEPTEMBER 2012

VIKRAM RAGHAVENDRA DESAI
B.E, VISWESWARAYA TECHNOLOGICAL UNIVERSITY,INDIA
M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Tilman Wolf

The introduction of programmability and dynamic protocol deployment in routers, there would be an increase in the potential vulnerabilities and attacks . The next-generation Internet promises to provide a fundamental shift in the underlying architecture to support dynamic deployment of network protocols. In this thesis, we consider the problem of detecting malicious packet drops 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 malicious packet detection technique that effectively detects malicious routers using delayed sampling technique and verification of the evidence. The verification involves periodically determining congestion losses in the network and comparing the forwarding behaviors of the adjoining routers to affirm the state of a router in the network. We provide a performance analysis of the detection accuracy and quantify

the communication overhead of our system. Our results show that our technique provides accurate detection with low performance overhead.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

The advent of network virtualization provides a practical approach to deployment and testing of various different protocol suite on network substrate [1, 29]. The virtualized network infrastructure facilitates in the development of new protocols in the current internet architecture. The dependence and limitations posed by independent physical infrastructure networks is overcome by virtualization in shared testbed. The virtual network architecture is composed of different entities, namely:

- Network Infrastructure: The network infrastructure provides the physical entities of the network. The task of the network infrastructure is to efficiently allocate network and data components among virtual networks, to ensure isolation of resource between each logical network.

- Virtual network: Each virtual network is composed of multiple virtual routers and network links which help deploy custom protocols on physical infrastructure leased from different network infrastructure.

The end users have the opportunity to choose multiple virtual network services. The architecture helps in deploying multiple logical networks on the same or different network infrastructure. Such a separation helps in improving scalability and reducing costs involved in setting up and maintaining networks.

The idea of deploying custom protocols on virtual networks provides for programmability in data-path of routers [11, 13, 32]. To accommodate on deployment of new custom protocols on virtual network slices, the virtual network substrate should

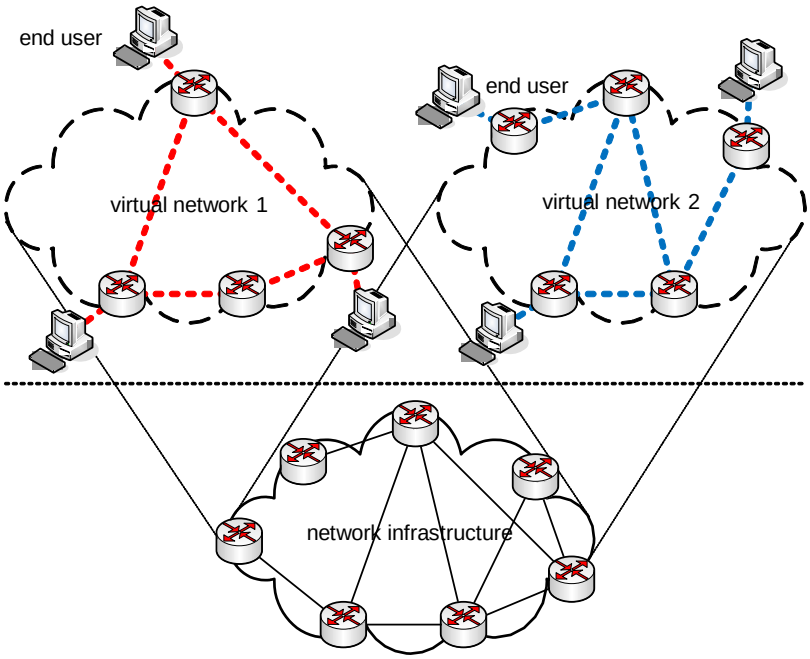


Figure 1.1: Virtual Network Infrastructure

provide custom packet processing. Provision of such facilities on the networks can be achieved by adding programmability in network processors or virtualization of the network operating system. To achieve programmability at the operating system level, the design needs to provide isolation between logical entities, performance and flexibility whilst keeping costs low. Research in this domain has suggested the use of building a set of well-tested and established plug-ins which can interact with hardware directly and each virtual network can be deployed by considering a subset of these plug-ins. As for processors, the recent developments in general purpose desktop processors have reached performance levels where they can handle high forwarding rates. This has led to the development of programmable packet processors as compared to application-specific integrated circuits(ASIC) whose functionality cannot be changed once they are designed. The process of adding programmability in network processors includes adding tasks to the threads of a processor. The threads have to handle packets that arrive at the ring buffer, call a user specified function to perform

packet processing and push the packet to the outgoing ring buffer once the function returns. The threads also need to do periodic tasks that are independent of packet arrival and should receive and respond to control messages. One of the issues of adding programmability of routers would be the advent of a whole new class of attacks that might be undertaken on the hardware.

As stated earlier, that customization of routers using programmability and virtual network infrastructure introduces increased vulnerabilities and attacks [7]. Routers based on general purpose processors are as vulnerable as end system hosts, because they contain the same vulnerabilities seen on the latter. On the contrary, the vulnerabilities in the network infrastructure pose an equal share for concern. Since virtual networks share the network infrastructure, any outage of service at the physical level would have catastrophic influence on the virtual networks. In addition to outages, if a back-end router, handling traffic up-to 40 Gbps is compromised, would lead to outage of service at multiple substrates. The attacker would not require physical access to the setup, but if he carefully crafts a packet containing malicious code, and executes the code on the router causing buffer overflow at the processor, leading to a stack smashing attack. The attacker can execute arbitrary operations on the packet processor, thereby causing a denial of service attack on the network infrastructure. It is extremely difficult to detect and isolate the particular router or group of routers that are compromised.

The attacks on network infrastructure take place by the combination of social engineering, weak passwords and software vulnerabilities. Current day routers command line interfaces are sufficiently powerful to drop and delay packets. It is extremely difficult to detect and isolate the particular router or group of routers that are compromised. Once the router has been compromised, the attacker can implement malicious attacks by selectively dropping packets, modifying the contents of the packet as well as routing packets to non-existent destinations (i.e., black hole

attack). In modern packet-switched networks, where traffic is forwarded by multiple routers that belong to different administrative entities, detecting misbehaving nodes is a challenge. One intuitive reaction would be to utilize traditional host-based intrusion detection techniques. The limitations of such an approach would be that once a router is compromised, the detection software cannot be dependable. It is also to be noted that recent mal-ware are capable of disabling the intrusion detection system. The network can be considered as point-to-point links connecting a pair of routers. Thus, the data must be forwarded hop-by-hop towards the destination. If a router along the path is compromised, it allows the attacker to drop, delay, corrupt or divert the packets passing through this router. The attacker would thus have the capability to deny service to legitimate hosts. There are predominantly two broad approaches of attacks that the adversary may undertake:

1. Control plane attack: The attacker may issue faulty routing advertisements and hence alter the network topology. This would disrupt services across the domain as the routers' view of the network topology is completely changed.

2. Data plane attack: The attacker would alter the forwarding action of the router, thus the router does not adhere to the routing table. The attacker would succeed in disrupting communication across the link.

The first set of attacks discussed has received a lot of attention due to it having a potential to disrupt services at a global level, but the second approach though not being entirely catastrophic, provides a lot more opportunity to an attacker to disrupt or degrade services by performing denial of service (DoS), packet modification, injecting new packets. We discuss about mechanisms that address the security issue, how to detect the existence of compromised routers in the network. For the case of a packet drop attack, it is even more difficult to determine if the packet drop was due to congestion losses or malicious behavior.

1.1 Contributions

The objective of this thesis is to design a secure controller based forwarding misbehavior detection system that uses a hash based detection delay sampling algorithm and a verification mechanism to detect the malicious packet drops introduced by routers.

Our main contributions are as below:

1. Formulation of the malicious packet forwarding router problem
2. Design a forwarding misbehavior detection system to accurately determine the malicious router
3. Implement the proposed design on Deter-lab for the proof of concept.
4. Presentation of a performance analysis of the proposed technique.

1.2 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 presents a detailed description of the general network and security model considered by most of the malicious router detection systems. Chapter 3 talks about the various systems developed as counter measures for attack on the packet forwarding mechanism. It presents the design decisions considered by the systems and the drawbacks faced as a result of them. Chapter 4 discusses the packet forwarding misbehavior detection design proposed and the advantages of the design compared to previous detection systems. The requirement for sampling and packet aggregation in achieving accuracy and reducing communication overhead. We also present a detailed description of the delay sampling approach used by the system. Chapter 5 presents the methods to determine congestion losses in the topology. Chapter 6 presents the analysis on the detection accuracy of the proposed system and the packet overhead incurred during detection.

Chapter 7 provides a summary and future work on the forwarding behavior detection system.

CHAPTER 2

PROBLEM SPACE

This chapter provides a background to understand misbehavior detection systems. We present a general model on which all of the previous misbehavior detection systems rely, and explore the general decisions made in their detection.

2.1 System Model and Assumptions

Most of the failure detection systems have similar requirements, such as a synchronous network model, correct terminal routers, property of a good path and cryptography. In this section, we present the requirements of failure detection systems for detecting compromised routers in the data plane.

2.1.1 Network Model

The misbehavior detection systems are designed for both hard-wired networks and wireless networks. Within the network we assume that the packets are forwarded by hop-by-hop manner by the routers [17]. The modern day routers are considered to have computational power to generate per packet or packet aggregate summaries for the traffic that it forwards as well as share these statistics with its neighbors. Most of the detection systems also require these routers to have synchronized clocks(e.g., The granularity achieved by NTP [14]).

The fault detection system also requires that there are sufficient different paths in the network such that more than two malicious routers do not partition the network. It is to be noted that most of the enterprise networks are designed with such path

diversities. However, this diversity is not extended to individual hosts, single workstations usually do not have multiple paths to their infrastructure. If a host's terminal router is compromised, the host is isolated and there is no way such a compromise can be detected. Since the traffic originates from a compromised router, we cannot detect anomalous forwarding behavior.

Most of the fault detection system that detect message corruption attacks require cryptographic functions. The cryptographic functions are primarily used for authenticity and integrity. We think that confidentiality is not a main concern for these protocols, if it is desired then the end systems are responsible to ensure confidentiality. Some of the techniques used are :

1. Digital signatures, e.g. DSA [27]
2. Message authentication code (MAC), e.g. HMAC [6]
3. One way hash functions, e.g. MD5, SHA-1 [31], UHASH
4. Pseudo random functions (PRF) [19]

In addition to cryptographic functions a key distribution mechanism is necessary. Key distribution can rely on either public or secret key infrastructures. It is assumed that either the ability to assign and distribute shared keys or a public key infrastructure is available. In our design, we make the assumption that the nodes can establish establish a secure connection between the controller and themselves (e.g., SSL [28]).

2.2 Detection Mechanisms

There are two approaches for detection of malicious nodes in a network. The Ack-based adversary identification and the distributed detection. We discuss about the approaches in this section.

2.2.1 Ack-based adversary identification(AAI)

The AAI detection mechanism [34] allows the source to monitor the forwarding path for packet dropping activity. Given a set of adversarial nodes located on a symmetric path, the source requires acknowledgment packets from the destination and the intermediate nodes along the path. Such a detection mechanism can only identify links adjacent to malicious nodes, rather than identifying the nodes. The AAI detection mechanism involves making decisions regarding (a) which data packets to acknowledge, and (b) which intermediate nodes should respond to the ack requests sent by the source.

2.2.2 Distributed detection

Distributed detection mechanisms [21] involve the neighboring routers in detecting compromised routers. A compromised router can potentially be identified by correct routers when it deviates from expected behavior. Such mechanism involve principles of traffic validation to help detect routers presenting anomalous behavior. For traffic entering a region of the network, and knowing the expected behavior of the routers in the network , anomalous behavior is detected when the traffic being monitored differs significantly from the expected behavior. Several characteristics of traffic can be summarized concisely and used to validate various monitored paths along the network. Some of the properties are:

1. Conservation of *flow* validates the volume traffic, thereby addressing the malicious behavior of dropping packets. Each router counts the number of packets that it has observed as it monitors traffic over some agreed time interval. Traffic validation is done by comparing values of these counters. The downturn of this technique is that it assumes that malicious nodes cannot fabricate packets to maintain the counters appropriately. However, it is extremely cheap to im-

plement in terms of per-packet cost and associated overhead to communicate traffic information among routers is low.

2. Conservation of *content* validates the content of the traffic , thereby addressing the malicious behavior of modifying packets. To detect modification of packets, a one-way hash value, of the payload can be used. Similar to the conservation of flow each router periodically distributes a set of hash values observed for traffic validation, which is calculated via set difference. One downside to this approach is that it requires storing and communicating hash value for each packet forwarded by the router.

3. Conservation of *order* validates the order among the packets that constitute the traffic, thus addressing packet reorder attacks. As with conservation of content, one way to detect packet reordering is to maintain ordered lists of packet hash values rather than just maintaining the hash values for packets observed. This can result in significant storage overhead.

4. Conservation of *timeliness* validates the time behavior of the forwarding process, thereby addressing the delay attacks on packets. Packet delay attacks can be detected by maintaining ordered list of packet hash values associated with timestamps. Traffic validation can be done by computing how much time is spent at each node for a given packet.

The goal of this thesis is to implement a intrusion detection system that is distributed in the network such that the detection can be performed using the existing hardware resources, thereby requiring the participation of non compromised networks. A compromised router can make alterations to its own forwarding behavior, however given that packet forwarding is distributed in nature, it is very difficult for the adversary to conceal such behavior. There is enough redundancy in the network to detect such alterations given that the packets traverse some non-compromised routers. The

process of synchronized collection of traffic information and distribution of these results leads to detection of compromised routers.

2.3 Security Model

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

2.3.1 Security Requirements

The security requirements ensure the secure processing and forwarding of traffic in the routers. The following are the security requirements for our network model:

- 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.
- Provide an inherent access control mechanism that protects from tampering the device.

2.3.2 Attacker Capabilities

A compromised router can behave arbitrarily by not participating in the protocol, announcing incorrect reports, or collude with other compromised routers to launch organized attacks. A *faulty* router is one that does not obey the protocol or alters the flow of traffic. The arbitrary behavior of a faulty router can be summarized with the threats listed below. When all these threats are non-existent, then no router is compromised.

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

- *Packet fabrication.* A compromised router can generate and inject arbitrary network traffic from the compromised router. This could be measured as the number of packets that are reported at the destination, considering these packets being sent from the source.

- *Packet reordering.* Reorder attack or Jellyfish attack is an attack in which a compromised node forwards data packets out-of-order

- *Packet modification.* A compromised router can corrupt the message field in the data packet. Such an attack may not be detectable by comparing the number of packets arriving at the destination from the source. Some summary of the content needs to be maintained, while one measure the number of packets.

- *Packet delay.* A compromised router can introduce arbitrary delay to multimedia traffic. Monitoring time stamps of the distributed aggregates can help detect such attacks.

CHAPTER 3

RELATED WORK

This chapter reviews some of the existing failure detection mechanisms. We study the various techniques used as countermeasure for the attacks on the network data plane. In Section 2.2 we talk about the detection techniques used by the various fault detection mechanisms, namely the *distributed detection* and the *Ack-based adversary identification*. Distributed detection techniques use traffic validation mechanisms to evaluate the performance of routers in a network. As defined in [22], there are two widely used approaches for validating traffic per path-segment:

1. *Validation at the terminal routers*: In Figure 3.1 the end nodes (namely the source and destination) along the forwarding path participate for validating the packets that traverse through the path. The end routers along the path exchange traffic information and validate traffic based on the same. The terminal routers 1 and 5 validate traffic through the path $< 1, 2....5 >$ and exchange traffic information. Such a mechanism does not detect compromised routers effectively, but it is widely deployed as it allows the nodes to synchronize on sampling and monitoring. Such a mechanism significantly reduces complexity

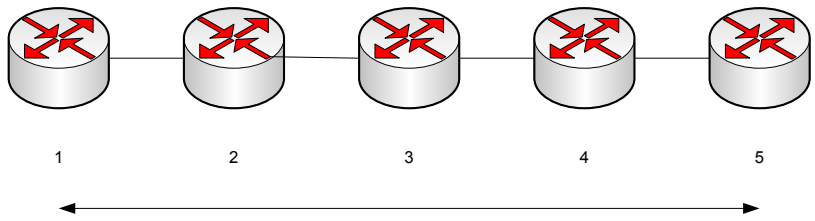


Figure 3.1: Traffic validation at the terminal routers

of the detection mechanism, though accuracy of detection is quite poor also communication overhead involves sending validation data at the either ends.

Detection mechanisms like *SecTrace* [25], *Byzantine Detection: PERLMAN* [26], *StealthProbing* [4], *Secure split assignment trajectory sampling* [18], *Availability centric routing* [30] apply this approach to detect anomalous behavior in networks. We believe that such a system in effective in detecting a compromised path, but does not satisfy the requirements of detecting compromised nodes. At best, such methods can detect a faulty path in the network. Also the accuracy of detection to the communication overhead achieved by such detection mechanisms is quite low compared to other techniques.

2. *Validation at each intermediate router* In Figure 3.2 All the nodes, the terminal routers as well the intermediate routers along the path participate in validation. All the routers along a given path, store information about the traffic they observe and exchange information periodically. Such a mechanism is ideal in detecting compromised router quickly and effectively. The downside of using such an approach would be cost of computation and communication overhead involved in exchanging traffic information. Our design uses a similar approach but we limit the traffic information exchange to three nodes and reduce the overhead of computation by performing detection at a centralized failsafe controller. Instead of all the nodes monitoring traffic, we schedule the traffic validation to the monitored node and its upstream and downstream neighbor. The accuracy of detecting a compromised node is high, so our detection mechanism would take slightly longer than the former method.

   Mechanisms such as *HSER* [3], *Early detection of message forwarding faults* [15] and *An on-demand secure routing protocol resilient to Byzantine failures* [5] implement traffic validation at each node. Such mechanisms are quick to detect

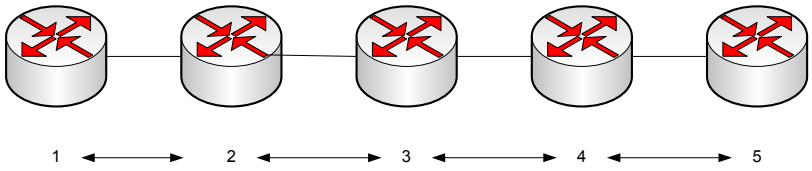


Figure 3.2: Traffic validation at the intermediate nodes

malicious behavior, but suffer from very high packet overhead. In our packet drop detection, we try to minimize this by having the controller a particular sequence of nodes at a given time. Schedule it to monitor three sets of node and thus, we tend to reduce the packet overhead and also improve on detection rate.

3.1 Validation at each intermediate router

In this section we discuss distributed detection mechanisms that employ the concept of validating traffic along every router. These mechanisms are highly sensitive in detection of attacks on network infrastructure but incur very high communication overhead in-turn.

3.1.1 Highly secure and efficient routing(HSER)

HSER [3] a mechanism that makes use of traffic validation per path segment nodes. HSER is a combination of source routing, hop-by-hop authentication, a priori reserved buffers, sequence numbers, timeouts, end-to-end reliability mechanisms, and fault announcements. The combination of these techniques provides Byzantine robustness and detection. HSER validates the conservation of content property of a single packet that is monitored along each path from source to destination. If any router along the path discovers that its neighbor has lost a packet, a failure is detected and an alarm is raised.

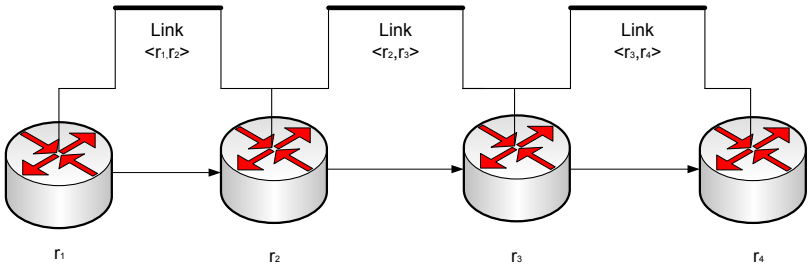


Figure 3.3: Highly secure and efficient routing

HSER requires the router to check for the authenticity of the packet and then forwards the packets to its downstream neighbor along the path. The authenticity can be verified by calculating a hash value over the packet and matching it with that generated by the upstream router. Once the router has forwarded the packet, it sets a timeout to the worst case round-trip time (RTT) to the destination. If however the timeout expires or the authenticity of the packet cannot be verified, the node then raises an alarm with a fault announcement that is sent to the source. HSER relies on source routing to act on its findings. Thus, upon receiving a fault announcement, the source router computes a new route to the destination excluding the detected link from its routing fabric. However, the overhead of this approach is quite high, since for every source and destination pair, all of the routers along the path must participate in the detection protocol. HSER tends to monitor a single packet at each round. There are thousands of thousands packets that must be forwarded in the network, and these mechanisms require some state of the packet being monitored such as a digest value or a time-stamp. Another limitation of such a design is that it does not consider benign packet losses. Modern routers routinely drop or delay packets due to congestion. The scheme also introduces a key storage overhead of O(d^2), for a *forwarding path* of length d.

3.1.2 Early detection of message forwarding faults

Herzberg et al. [15] present a mechanism of early termination to detect Byzantine failures. The early termination is a specification to deliver a message from a processor u to a processor v, and to detect anomalous behavior if there exists a fault along the communication path in minimal time and low communication overhead. The detection mechanism is based on acknowledgments and use a time-out value D, which depends on the transmission delay over the communication path. The authors present various mechanisms to detect failures using acknowledgments from destination node as well as some intermediate nodes to the source. Usually such failures are detected by hop-by-hop acknowledgments, which are sent from a node to its neighbor on receiving a message from it. There are some failures which cannot be detected by such a mechanism, e.g., a neighbor node may send an acknowledgment without forwarding the message downstream. Such a failure might take place even when a node breaks down after sending the acknowledgment, such failures cannot be detected by hop-by-hop authentication. Rather you would need an end-to-end mechanism to detect such a failure.

Initially the authors propose a $end - to - end$ fault detector, wherein a destination receives a packet, it sends back an acknowledgment along the same path. Every intermediate node i checks to see if node $i + 1$ is faulty. The node expects to receive an acknowledgment from its neighbor, and maintains a timeout clock for the same. If the node does not receive an acknowledgment or a disconnect request before the expiration of the timer, node i detects $i + 1$ faulty and sends an announcement to the source determining the path $< i, i + 1 >$ as faulty. The communication overhead involved is less, since it only involves a single message. Such a mechanism however suffers from very high time complexity to detect failures on the data path. To improve on time complexity in detection of failures, they tried a hop-by-hop fault detection where in a node would send an ack after forwarding the data packet and maintains a

timeout for intermediate nodes to the destination. In this mechanism, the detection of faulty link can be done optimally but the communication overhead involved is very high since all the intermediate nodes participate in the detection process. Thus the authors came up with an optimal detection mechanism wherein some of the intermediate nodes send ack to pre-assigned nodes along the path and by assigning the optimal number of intermediate nodes, they were able to achieve optimal detection time and message overhead.

3.1.3 Fatih

Fatih [22] implements distributed detection using the traffic validation techniques as discussed in the previous chapters. The detection technique requires a consensus among the routers of a particular path about the traffic information to be distributed and applying traffic validation in its detection. Each router monitors a set of path segments, which contains all of the $k+2$-path segments containing the router and all the x-path segments, such as $3 \leq x < k+2$ whose ends terminal routers. The routers synchronize among themselves for each path segment and collect information about the traffic for a agreed upon time interval t. Each router then applies traffic validation to the data received. The detection mechanism implemented by Fatih is on similar lines as discussed in the previous chapter, though it implies considerable packet overhead in terms of traffic information shared by the routers as well as for synchronization. Moreover, such a detection mechanism can at best detect a faulty path, but cannot help come to a consensus as to which router along the path is faulty and presenting malicious behavior.

3.1.4 An on-demand secure routing protocol resilient to Byzantine failures

[5] present an on-demand routing protocol for ad hoc wireless networks that provides resilience to byzantine failures caused by a single malicious node or colluding

nodes. The authors claim that for wireless environments, on-demand routing protocols are more appropriate as they initiate a route discovery process only when data packets need to be routed. The authors divide their on-demand routing protocol detection mechanism into three phases, namely *route discovery with fault avoidance*, *byzantine fault detection*, *link weight management*. Route discovery with fault avoidance entails in finding the least weight path from the source to destination. Byzantine fault detection takes the data path between the source and the destination as input and outputs a faulty link.

The fault detection algorithm assumes a loss threshold that sets a bound on the loss rate. When the losses observed is greater than or equal to the threshold, the detection system terms the state of the path as faulty. Since the value of the threshold also determines the amount of loss the adversary can cause without going detected, the threshold value should be as low as possible, still greater than the congestion loss rate. The detection mechanism requires the destination to send on ack for each received packet. The source monitors the losses and when the loss rate on a path observed is greater than that of the threshold, the source starts a binary search on the path. The search is designed in way that intermediate nodes, with the destination are now required to send acks back to the source. These intermediate nodes are termed as probe nodes. Since the list of probes is injected with the traffic, the compromised node must also drop the probe list. Once the compromised nodes list is dropped, it is easy to detected the compromised path between probe nodes. The probes divide the path into non overlapping sub-paths, then a new probe is added between the path between the two probe and it is added to the active probe list maintained by the source node. This continues till the fault is detected on a interval that forms a single link between two probe nodes. This link is sent to the link weight management. The binary search proceeds by one step on each fault detected, this leads to detection of the faulty link after $log\ n$ faults are observed. The drawbacks involved in this approach is that it can

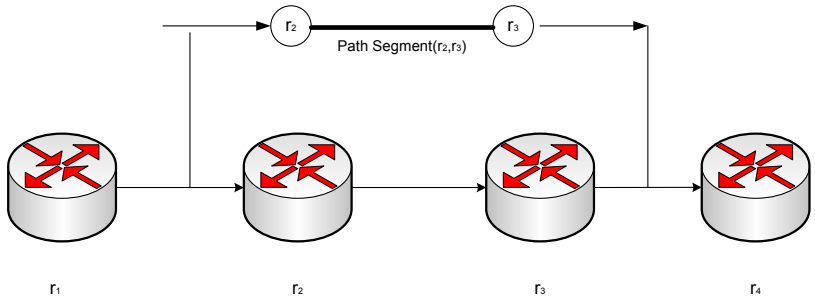


Figure 3.4: Secure traceroute

only determine the faulty link, but not point at the compromised router. Detection time for long paths are quite high as you would need *log n* faults to detect a faulty link.

3.2 Validation at the end nodes

In this section we describe detection mechanisms that involve only the end nodes . Such techniques have a lower accuracy of detection compared to the former but have considerable lower communication overhead.

3.2.1 Secure Traceroute (*SecTrace*)

SecTrace [25] was developed to be a practical tool for securely tracing the path of traffic from a source toward a particular destination. As in the case of normal traceroute, SecTrace proceeds hop-by-hop expecting each node to respond to the traceroute. The node sends an identification marker for the packet as a response to the tracing node.

In the above figure 3.4 path segment of r1, r2, r3, r4 is monitored during the given traffic validation round, and only the source r1 and the corresponding intermediate router r4 implement the distributed failure detector. If r1 detects a discrepancy in the traffic, a failure is detected and an alarm is raised. Either one of the intermediate routers r2, r3 is traffic faulty, introducing discrepancy into the monitored traffic, or

the failure detector, which is implemented by r1 and r4, is detection protocol faulty: at least one of r1, r4 is faulty. The accuracy of prediction is in the assumption that malicious actions are consistent over time and are not Byzantine. Such a technique at best can detect if a monitored path is faulty and its only at the source that detection of malicious activity takes place. The mechanism has a high packet overhead as it requires the intermediate nodes to respond with ACK's to the source request. The authors suggest that the detection process can be started periodically rather than detection taking place at all times.

3.2.2 Byzantine detection: PERLMAN

In her thesis, Perlman [26] presented the ideas of Byzantine robustness and Byzantine detection. She talks about a method of *robust flooding*, which ensures that a packet is delivered to all properly functioning routers. Such a process would require a *good path condition*, which states that each pair of non-faulty routers is connected by at least one path of non-faulty routers. such techniques were used for public key distribution. The drawback of this mechanism is that if a path is detected as faulty, it treats all the routers in the path as faulty and avoids sending data through those path.

3.2.3 Stealth Probing

Stealth Probing [4] is a end-router-to-end-router secure data plane monitoring mechanism. It determines whether end router paths are operational even amidst attacks on the network infrastructure. Stealth probing employs an *IPSEC* [16] tunneling between two end routers and transmits both the data packets as well as probe ICMP packets through the tunnel. In such a scenario, the attacker cannot adaptively drop only the data packets without dropping the probe packets as well. Hence it would be very difficult for the attack to go unnoticed. The mechanism of introducing probe packets is termed as active probing mechanism. The authors also talk about a passive

probing mechanism wherein the tunnel entry and exit points agree upon an agreed hash function to be applied on the immutable field of each packet prior to encryption and after decryption. The tunnel exit router should acknowledge any discrepancies in the hash digest values. This technique is similar to *Trajectory Sampling* [10].

3.2.4 SATS

Secure split assignment trajectory sampling [18] (SATS) builds upon the idea of *Trajectory Sampling*. Trajectory sampling [10] involves each router calculating a hash value over the packet content that does not change along the flow path. The packet is sampled if the resulting hash value falls in a predetermined range. The packets sampled by each router is synchronized as identical hash functions and ranges are used. Each packet that is sampled, the router computes a fingerprint of the packet and reports to a controller node.

In SATS though, the controller node assigns different hash ranges to every pair of routers in the autonomous system. The controller node uses encrypted channels to assign hash ranges to the pair of routers. This minimizes the chances of the attacker to evade detection. Each router monitors packets and sends report to the centralized controller node. If the controller node detects a inconsistency between the reports of the pair of routers, then all the routers between the reporting routers including the reporting routers are suspected to be compromised. As with trajectory sampling, the problem with SATS is that the compromised node may provide preferential treatment to sampled packets and drop the other packets. Such an act from a compromised router may hide details about its actual performance.

3.2.5 ACR

Availability centric routing (ACR) [30] provides secure inter-domain routing with promise of end-to-end confidentiality, integrity and availability. The mechanism assumes that multipath are available through a custom protocol or with the proposed

add-path extension to BGP [8]. The goal of ACR is to allow end systems to securely communicate with each other even though some parts of the network infrastructure may be compromised by the adversary. The detection mechanism assumes that the intermediate ASes should provide multiple routes to the edge router for the destination. Sources should verify that the destination is accessible through the chosen route. The ACR end systems securely communicate with each other and monitor the performance along the path. If the performance is not satisfactory the end systems signal ACR to use a different path. The end systems also distribute traffic over one or more paths supplied by the intermediate ASes by applying selection algorithm to quickly identifying working paths with high probability.

CHAPTER 4

PACKET FORWARDING MISBEHAVIOR DETECTION

Current distributed monitoring approaches involve mostly the terminal routers in the forwarding path to detect malicious routers. This introduces a significant communication and storage overhead to validate the routers. Hence we propose a controller-based monitoring technique that gathers forwarding behavior information from neighboring nodes of the monitored node. This technique introduces lower performance overhead as well as improves the detection accuracy as only a subset of the routers along a path of a network are involved. The assumptions of our network model is similar to those described in subsection 2.1.1.

4.1 Design

The technique of using witness-based forwarding misbehavior detection in wireless networks was shown in [33]. Unlike the approach employed by the witness-based model, which chooses the witness nodes from the neighboring nodes that overhear a node's transmission, we consider a secure controller node (for a given autonomous system) that dynamically picks and collects sample aggregate from the node being monitored and evidence from its neighbor nodes.

The design of the forwarding misbehavior detection system is as shown in Figure 4.1 [9]. The system consists of a controller node (C, that does not lie on the forwarding path of the router), 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 the Figure 4.1.

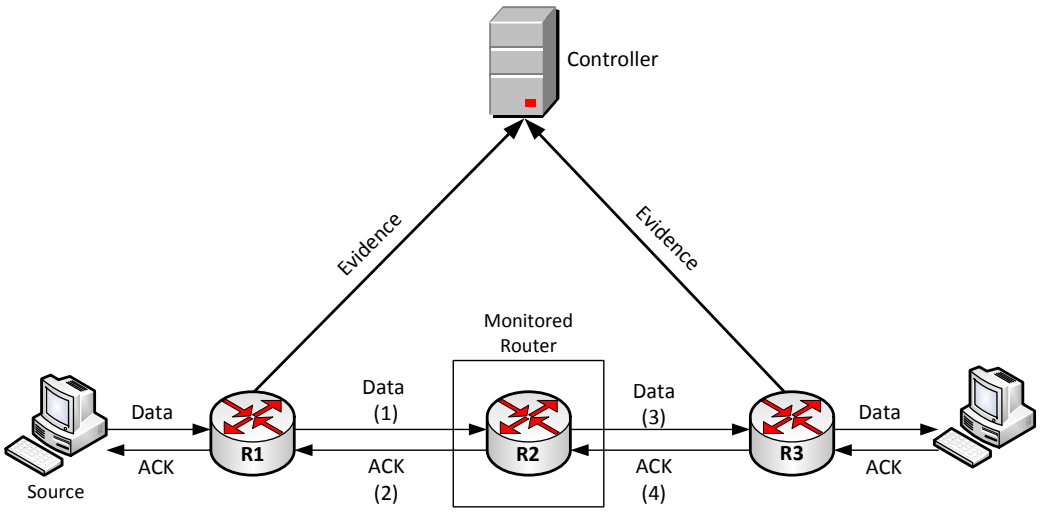


Figure 4.1: Packet Forwarding Misbehavior Detection

At a random time interval t, the controller node C monitors the forwarding behavior of router R (We assume that nodes U and D are secure at that instance of time). For effective monitoring, routers U and D together send sampled evidence statistics to C. The controller then performs a verification mechanism on the received packets to determine the state of the monitored node R.

4.2 Detection Mechanism

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

4.2.1 Challenges

The fundamental challenge of our misbehavior detection technique is to provide an accurate detection of the state of router R at a given period of time.

How to monitor packets of a flow is another important design decision, whether to monitor a single packet or aggregate traffic. Some of misbehavior detection systems such as HSER monitor a single packet at each round. There are thousands and thousands of packets that must be forwarded in the network, thus it is required to

maintain state,such as timestamp, hash-value etc for each packet observed in the network. This results in a explosion in the state size required at each router if we monitor all the packets in the network. Also, if we monitor state for each packet, its quite hard to attribute missing packets to malicious drops as modern day routers drop packets when the load exceeds their buffering capability. So if a monitored router tends to drop packets due to congestion, then it may be incorrectly identified as faulty. In contrast, SecTrace, validate aggregate traffic over some period of time. While these protocols compute some state locally for the traffic, doing it over an aggregate of traffic reduces the communication and synchronization overhead across the network. Validating over a traffic aggregate makes it possible to apply a threshold mechanism to distinguish between benign losses and malicious behavior. Considering the aggregation of traffic technique, monitoring all the packets in the aggregate might still be costly. One can easily trade-off accuracy for packet overhead by sampling the packets to be considered. If the monitoring application needs to relate packets coming from different measuring points, one must ensure that the same set of packets are selected at each measuring point. The process of choosing the same set of samples is termed as consistent selection. Thus, to address the above challenges, the proposed design mechanism involves 1) a robust sampling technique and 2) a verification technique that is effective in detecting the malicious packet processing behavior.

4.2.2 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 [2]. Delay sampling requires

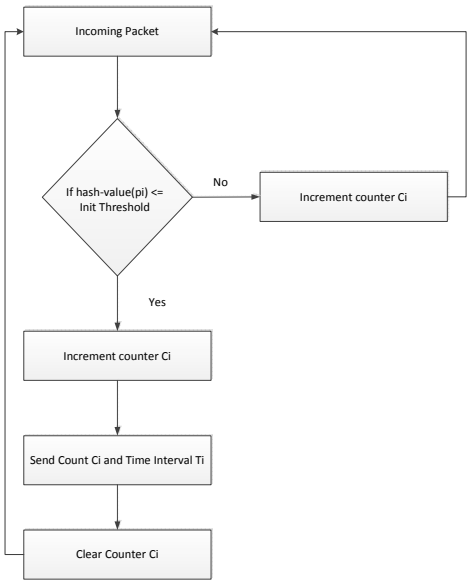


Figure 4.2: Sampling Flow

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 [2] to support the monitoring of multiple flows (maintaining individual counters for each flow) at a given time. For each flow $flow_i$, the counter maintains a count of the observed packets. 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 hash value is below a predefined threshold. On receiving the initiator packet, U and D sample the evidence packets and send the count of samples, the time-stamps of the previous initiator packet, current initiator-packet to the controller node for verification.

Table 4.1: Sampling Details

Variables	Definition
p	new packet
$InitiatorID_i(p)$	hash function
η	initiator threshold
C_i	Packet counter for a path i
T_i	Interval between the initiator packets

The set of sample variables and the associated definitions used in the sampling algorithm are shown in Table 4.1. Each node associates a packet with a $FlowID$ that defines the packet's flow and computes hash functions ($InitiatorID$). The node also maintains k counters (C_i) for the recently seen k unique flows and k intervals between initiator packets (I_i), which are sent to the controller node. The process of maintaining k counters at a given instance of time improves the detection accuracy compared to observing a single stream of packets. The creation and deletion of the tuples are based on the k recent flows observed in the router. For simplicity, we show the packet sampling process for a single flow.

Algorithm 1 shows the sampling algorithm when a new packet is received. First the node computes (for flow i) the $InitiatorID_i$ (p) for the received packet.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, corresponding counter C_i is updated with packet p.

Next, if the hash value of the packet is less than that of the set $InitiatorID_i$. The count in counter C_i and the time interval between two initiator packets T_i is sent to the controller node for verification.The sampling method is initiated at the uplink and downlink routers and evidence sent to the controller node for verification. The initiator packet ensures that the same packet counts are sent at each node.

Algorithm 1 Sample <p>

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

1: **for** $i = 1$ k **do**
2: **if** $InitiatorID_i(p) \leq \eta_i$ **then**
3: $S_i \leftarrow T$ $\{Add\ the\ count\ and\ time\ interval\}$
4: $Clear\ C_i$
5: $Clear\ T_i$
6: **else**
7: $C_i \leftarrow p$
8: **end if**
9: **end for**

end

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 ($InitiatorID_i(p)$). The probability of seeing an initiator packet after p is given by the cumulative distribution function (CDF) as:

$$CDF = 1 - (1 - \frac{\eta}{M})^n \tag{4.1}$$

4.2.3 Verification

The verification is performed at the controller node C when it receives the evidence from the uplink (U) router and the downlink (D) router.The sampling rate (S_{rate}) of the uplink and downlink nodes are calculated by considering the packet count against the time interval between initiator packets. Congestion losses (C_{loss}) (Section 5.2) for $path_i$ are determined at downlink router (D), and updated at the controller (C) periodically. The exponential moving average of the sampling rate of each node is

determined at the controller node when samples are received.The exponential moving average is determined as :

$$S_{avg} = \alpha \times S_{rate} + (1 - \alpha) \times S_{avg}$$

$$0 \leq \alpha \leq 1 \tag{4.2}$$

The forwarding state of the router is determined by considering the sampling rate of downlink node (D) against that of the sampling rate of the uplink node (U). This is compared with the forwarding threshold determined by utilizing the congestion losses detected, the forwarding threshold would determine the sensitivity of the malicious router detection.The state of the router would be detected as follows:

$$\frac{S^D_{avg}}{S^U_{avg}} < \gamma \times (1 - C_{loss})$$

$$0 \leq \gamma \leq 1 \tag{4.3}$$

The value of γ chosen by the network, would determine the threshold that could be used to confirm the state of the router. If the forwarding quotient of the router is less than the determined forwarding threshold, the state of the router is determined to be *bad*, else the router is said to be in a *good* state.

CHAPTER 5

INFERRING CONGESTION LOSSES

We are concerned with detecting if a router is maliciously treating the packets that it observes. In case of a malicious router selectively dropping packets, it is very difficult to term missing of packets to malicious behavior, as normal congestion in todays network can bring about similar behavior at the routers. Routers drop packets when the inflow of packets exceeds their buffering capacity. Transmission control protocol (TCP) causes such losses as a part of its congestion control behavior.

5.1 Methods to Determine Congestion Loss

In building a malicious packet drop detection technique, its very important to determine whether the loss occurred due to congestion or due to the router being compromised [23]. The methods used to determine the nature of packet losses are as follows:

1. *Static Threshold.* The static threshold is a user defined threshold, which suggests that if the determined losses in the network is less than the threshold value, then the router is not compromised. Too many losses would mean malicious intent. One fundamental question is how to choose a value to set as the threshold. Depending on the threshold set, the system may produce false positives for benign losses or produce false negatives by failing to detect malicious losses caused by highly focused attacks. In order to avoid false positives, the threshold must be large enough to encompass all possible benign losses. However, this would lead an attacker to cause packet drop rate up-to the set

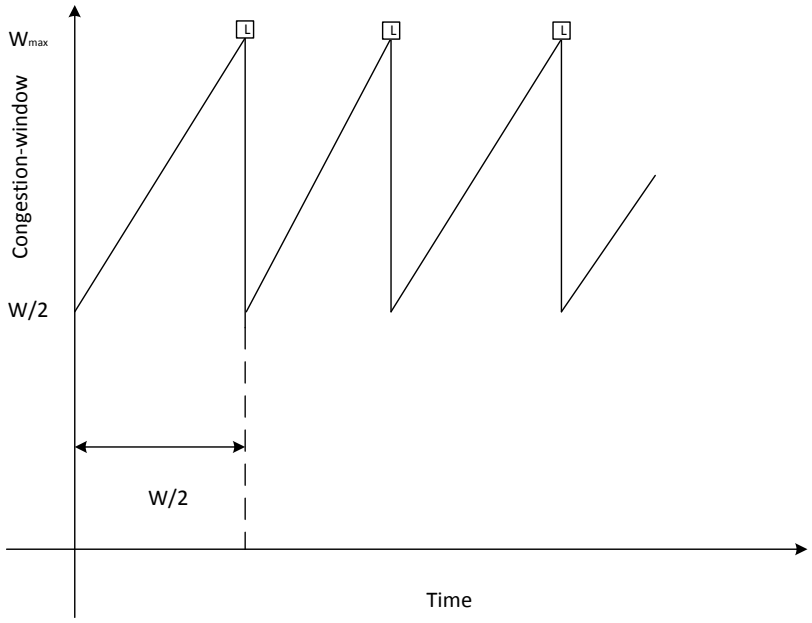


Figure 5.1: TCP Saw-tooth Behavior

threshold, thereby going unnoticed. Considering the vast usage of transmission control protocol (TCP), in todays network and that TCP reacts even to a small number of packet losses, the static threshold technique is not efficient as it allows attackers to impact performance without being detected.

2. *Traffic Modeling.* Packet losses are calculated based on the measured values of the traffic parameters and losses greater than the calculated loss are termed to be malicious. This can be achieved by determining congestion as function of the individual flow parameters. In our detection mechanism, we have chosen to determine benign losses in a network using network traffic modeling. This approach helps in removing the ambiguity on packet losses, as we can compare the measured losses with the loss rate determined by the congestion function.

5.2 Modeling Congestion Losses

We consider a TCP flow in congestion avoidance mode. In our congestion analysis model, we do not consider the slow start phase as these are typically short and the sender grows out of the phase exponentially [17]. The congestion window varies from $\frac{W}{2}$ to W_{max}, a single packet is dropped when the congestion window size reaches W_{max}. In the TCP congestion avoidance phase, we know that congestion window increases by one segment (MSS) per round trip time. Therefore the time for network to drop a packet for the TCP flow is

$$Time = \frac{\frac{W}{2}}{MSS} \times RTT \tag{5.1}$$

From the TCP throughput (T) formula, we can deduce that:

$$W_{max} = T_{max} \times RTT \tag{5.2}$$

$$T_{max} = \frac{4}{3} \times T_{avg} \tag{5.3}$$

The above equations help in determining the time required for the network to drop one packet when the maximum size of the congestion window is observed. When we consider the time to drop one packet over the duration that we monitor the network, we can deduce the total number of benign losses observed over the link. The number of lost packets were compared with the results of a TCP probe captured on the sender's side to verify the calculated losses. Thus, we obtain the congestion loss probability by comparing the count of the number of packets lost to the number of packets injected into the network.

This model is similar to the model suggested by [20] , which places an upper-bound on the congestion loss probability (p) by the following formula:

$$p < \left(\frac{MSS}{T_{avg} \times RTT} \right)^2 \tag{5.4}$$

In the malicious packet drop detection mechanism we periodically calculate the congestion loss probability as shown in Equation 5.4 and maintain a mean congestion loss probability observed over time. As described in the subsection 4.2.3, the detected loss probability is used to determine the state of the router. This stochastic congestion loss model, though being highly idealized, can be utilized to model a loss threshold for individual flows being monitored.

CHAPTER 6

EVALUATION

6.1 Experimental Setup

As a proof of concept we have implemented the malicious packet drop detection mechanism on Deterlab testbed [24]. In our experimentation, we used the simple topology as shown in the figure 6.1. The routers composed of a single 3.0 Ghz 64-bit Xeon processor, 2 GB of RAM and are running Linux operating system. The uplink and downlink nodes form connection with the controller node. These links each have a bandwidth of *1Gbps*. The links connecting the terminal routers to host nodes are also *1Gbps* each. The link connecting node2 (R) and node3 (D) has a bandwidth of *10mbps*. This link acts as a bottle neck and hence we will observe congestion in the network. We can utilize this to calculate congestion losses and then compare them with the total losses observed over a link to determine if any router was malicious. The source node generates three TCP flows to the destination.

We utilize *Libipq* to obtain the five-tuple packet header details. Libipq is a development library for iptables userspace packet queuing. The protocol header information can be obtained to build a tuple for each packet observed at the routers. Once the tuple for each packet is obtained, the packet can be put back into the kernel with a decision as to forward the packets or to drop them. Once a tuple is obtained for a packet, we create a message digest (SHA-1 [31]) on the packet tuple information that is obtained.We use the Hashlib++ library to generate the message digest. SHA-1 gives out a forty character hexadecimal output, of which we consider four contiguous characters to obtain a binary representation. We later obtain a decimal value of the

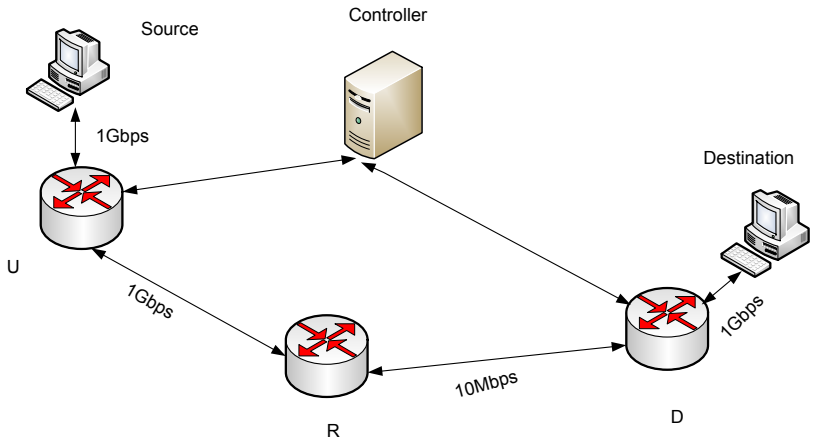


Figure 6.1: Topology

binary representation of the chosen characters. If the decimal value is lesser than a sample initiator threshold, the count of packets is sent to the controller from uplink U and downlink D routers. Using a similar approach, we can vary the drop rate threshold on router R for a particular flow. The controller deploys a verification algorithm for samples received of each flow and detects the forwarding ratio, hence classifying whether the state of the router is good or bad. We alter the drop rate in router R , and compare the average sampling rates of the downlink and uplink routers to determine the drop rate of monitored router R. We compare the determined drop rate with the congestion loss calculated as shown in Section 5.2 and employ the verification technique as discussed in the subsection 4.2.3.

6.2 Results

In this section, we discuss the performance analysis of our proposed technique. 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 state of the router, and 2) lower performance overhead. We provide an analysis for the above two performance metrics.

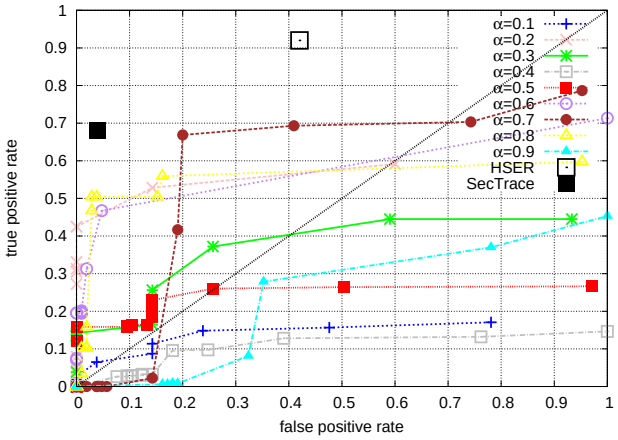
6.2.1 Detection Accuracy

To verify the accuracy of detection of the packet drop detection technique, we determine the malicious behavior by varying the attack rate. We monitor the node for 120 seconds, in which the node is in a good state, i.e., it does not cause malicious drops for the first 30 seconds and drops packets for the remaining 90 seconds. Thus, we observe that the monitored node (R) stays in a good state for *25%* and for the rest *75%* the router drops packets for a set rate. The parameters set in the controller for detection include the exponential smoothing factor α, used in determining the average sample rate of each node.

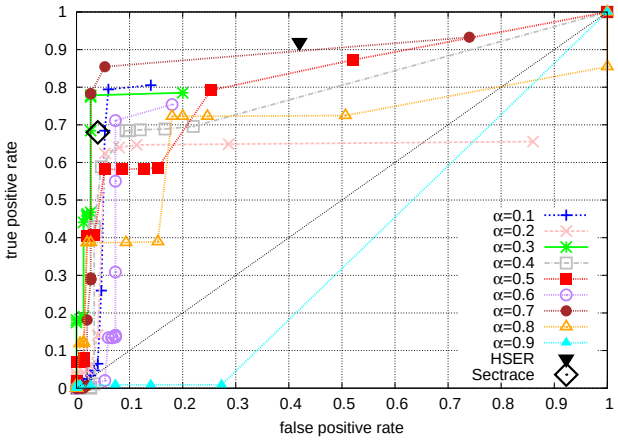
In equation 4.2, we calculate the exponential weighted moving average of the sample rates for the samples received from node U and node D. The α used in the determining the exponential weighted moving average is termed as the smoothing factor. The smoothing factor determines the reactivity of the model. When α is close to 1, the averages are influenced by the latest values and hence the model is over-reactive. When α is close to 0, the model is less reactive and the values of the former periods have a larger influence on the moving average.

In the malicious packet drop detection mechanism, the node verification is done comparing the forwarding rates of the uplink and the downlink nodes. As discussed before, the sampling averages determined are dependent on the value of the smoothing factor α. We empirically determine the value of α in our mechanism, as it allows to fine tune the sensitivity of our detection scheme. If α is chosen to be very close to 1, there are high false positives observed, whereas for lower values of α the false negatives increase as it takes a long time to determine the change in the state of a router. Thus an optimal value of the smoothing factor (α) is to be determined, which would result in lower false positives and false negatives.

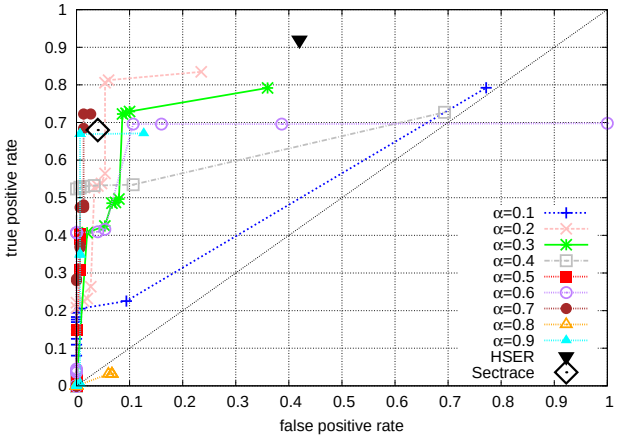
Determining the optimal value of α has to be performed in an empiric way. We apply the detection mechanism on a series of prior known states of the router. While



(a) ROC curve for 1Mbps



(b) ROC curve for 10Mbps



(c) ROC curve for 1Gbps

Figure 6.2: Smoothing Factor for Different Flows

Table 6.1: Optimum Values for Smoothing Factor(α)

Bandwidth	α	Youden Index
1Mbps	0.8	0.4754
10Mbps	0.7	0.7002
1Gbps	0.2	0.6459

applying the detection mechanism on the training states, we vary the value of the smoothing factor(α) from 0 to 1. We compare the detected states against the training state and plot the receiver operating characteristic (ROC) curves. An ROC curve is an illustration of the sensitivity of a binary classifier (i.e., the packet drop detection mechanism) [12]. The ROC is a plot of the true positive rate (TPR) against the false positive rate (FPR). The ROC curve where the distance between the curve and the diagonal is maximum would provide an optimal value for the smoothing factor (α). This is determined by calculating Youden index J. Youden index J is the point on the ROC curve which is farthest from the line of equality (diagonal line). Youden index is determined as max(TPR+(TNR-1)). As seen in Equation 4.3 , we vary the value of γ from 0 to 1 and obtain the true-positive rate and false-positive rates for the different thresholds. The Youden index value helps determine the optimal values of smoothing factor for flows with different bandwidth delay product($1Gbps$,$10Mbps$ and $1Mbps$).

Figure 6.2 shows the ROC curves for value of smoothing factor varied from 0 to 1 for flows with bandwidths of $1Mbps$, $10Mbps$, $1Gbps$. We have to consider line plots in the ROC curve that are closer to the top left corner of the graphs. As seen in Table 6.1 we observe that for a bandwidth of *1Mbps* the optimal value for smoothing factor(α) to be *0.8*, similarly we observe from figure 6.2 the optimal value for *10Mbps* and *1Gbps* to be 0.7 and 0.2 respectively. Hence using empirical technique we can similarly determine the optimal value of the smoothing factor (α) for flows with different bandwidth and delay.

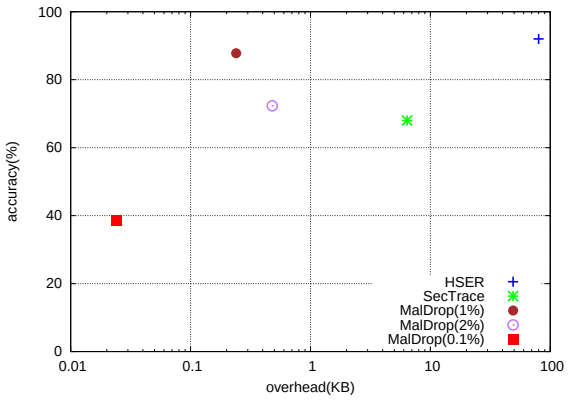


Figure 6.3: Accuracy vs Overhead

Table 6.2: Communication Overhead comparison for 1000 packets

Technique(threshold)	overhead (KB)	Accuracy(%)	Smoothing Factor(α)
MalDrop(0.1%)	0.024	38.69	0.1
MalDrop(1%)	0.24	87.7	0.7
MalDrop(2%)	0.48	72.3	0.5
SecTrace	6.4	68	
HSER	80	92	

6.2.2 Performance Overhead

The total packet overhead in the detection mechanism is determined by two factors, the count of the number of samples and the time difference between sampling intervals. If we utilize an unsigned integer (4 bytes) to store the packet count and a double (8 bytes) to store the time difference between the sampling initiator packets, the size of evidence sent to the controller node is about 12 bytes. Since the uplink (U) and the downlink (D) node sample, the overhead is twice of the data sent to the controller from one node. The tradeoff between detection accuracy and communication overhead is shown in figure 6.3. The overhead involved when the threshold ($\frac{\eta}{M}$) is 0.001 is less as the sampling is initiated on an average of 1000 packets as compared to the sampling threshold of 0.02, wherein sampling is more frequent at sampling being initiated at 50 packets. We notice that with sampling initiated on

every 100 packets, we observe detection accuracy comparable to that of HSER with considerable reduction in comunication overhead.

CHAPTER 7

SUMMARY AND CONCLUSION

In this thesis, 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's corresponding uplink and downlink nodes. The performance analysis of our proposed mechanism shows the introduction of lower packet overhead compared to HSER and SecTrace. We also achieve effective detection accuracy with lower false positive rates. Currently, our design assumes no collusion among the monitored node and the adjoining nodes. In the future, we should consider collusion among malicious nodes as a factor in detection. The concept can be extended to involve other form of attacks such as packet fabrication, packet modification and packet delay.

BIBLIOGRAPHY

[1] Anderson, Thomas, Peterson, Larry, Shenker, Scott, and Turner, Jonathan. Overcoming the Internet impasse through virtualization. *Computer 38*, 4 (Apr. 2005), 34–41.

[2] Argyraki, Katerina, Maniatis, Petros, and Singla, Ankit. Verifiable network-performance measurements. In *Proceedings of the 6th International COnference* (New York, NY, USA, 2010), Co-NEXT '10, ACM, pp. 1:1–1:12.

[3] Avramopoulos, Ioannis, Kobayashi, Hisashi, Wang, Randolph, and Krishnamurthy, Arvind. Highly secure and efficient routing. In *Proceedings. IEEE INFOCOM 2004, HONG KONG* (2004).

[4] Avramopoulos, Ioannis, and Rexford, Jennifer. Stealth probing: efficient data-plane security for ip routing. In *Proceedings of the annual conference on USENIX '06 Annual Technical Conference* (Berkeley, CA, USA, 2006), USENIX Association, pp. 25–25.

[5] Awerbuch, Baruch, Holmer, David, Nita-rotaru, Cristina, and Rubens, Herbert. An on-demand secure routing protocol resilient to byzantine failures. In *ACM Workshop on Wireless Security (WiSe* (2002), pp. 21–30.

[6] Bellare, Mihir, Canetti, Ran, and Krawczyk, Hugo. Message authentication using hash functions: the HMAC construction. *CryptoBytes 2*, 1 (Spring 1996), 12–15.

[7] Chasaki, Danai, Wu, Qiang, and Wolf, Tilman. Attacks on network infrastructure. In *Proceedings of Twentieth IEEE International Conference on Computer Communications and Networks (ICCCN)* (Maui, HI, Aug. 2011).

[8] claffy, k. Border Gateway Protocol (BGP) and Traceroute Data Workshop Report. Tech. rep., Cooperative Association for Internet Data Analysis (CAIDA), Oct 2011.

[9] Desai, Vikram, Natarajan, Sriram, and Wolf, Tilman. Packet forwarding misbehavior detection in Next-Generation networks. In *IEEE ICC 2012 - Communication and Information Systems Security Symposium (ICC'12 CISS)* (Ottawa, Canada, June 2012).

[10] Duffield, N. G., and Grossglauser, M. Trajectory sampling for direct traffic observation, 2001.

[11] Dutta, Rudra, Rouskas, George N., Baldine, Ilia, Bragg, Arnold, and Stevenson, Dan. The SILO architecture for services integration, control, and optimization for the future Internet. In *Proc. of IEEE International Conference on Communications (ICC)* (Glasgow, Scotland, June 2007), pp. 1899–1904.

[12] Fawcett, Tom. An introduction to ROC analysis. *Pattern Recognition Letters 27*, 8 (June 2006), 861–874.

[13] Feldmann, Anja. Internet clean-slate design: what and why? *SIGCOMM Computer Communication Review 37*, 3 (July 2007), 59–64.

[14] Gurewitz, Omer, Cidon, Israel, and Sidi, Moshe. Network classless time protocol based on clock offset optimization. *IEEE/ACM Trans. Netw. 14* (August 2006), 876–888.

[15] Herzberg, Amir, and Kutten, Shay. Early detection of message forwarding faults. *SIAM J. Comput 30* (2000), 1169–1196.

[16] Kent, S., and Atkinson, R. Security architecture for the internet protocol, 1998.

[17] Kurose, James F., and Ross, Keith W. *Computer Networks*, 3rd ed. Addison Wesley, 2004.

[18] Lee, Sihyung, Wong, Tina, and Kim, Hyong S. Secure split assignment trajectory sampling: A malicious router detection system. In *Dependable Systems and Networks* (2006), pp. 333–342.

[19] Luby, Michael George, and Michael, Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, Princeton, NJ, USA, 1994.

[20] Mathis, Matthew, Semke, Jeffrey, Mahdavi, Jamshid, and Ott, Teunis. The macroscopic behavior of the TCP congestion avoidance algorithm, 1997.

[21] Mizrak, Alper T. *Detecting Malicious Routers*. PhD thesis, University of California, San Diego, Sept. 2007.

[22] Mizrak, Alper Tugay, Cheng, Yu-Chung, Marzullo, Keith, and Savage, Stefan. Detecting and isolating malicious routers. *IEEE Transactions on Dependable and Secure Computing 3*, 3 (Jul-Sep 2006), 230–244.

[23] Mzrak, Alper T., Marzullo, Keith, and Savage, Stefan. Detecting malicious packet losses. Tech. rep., International Conference on Dependable Systems and Networks, 2007. January 2007 ALPER T. MIZRAK Page 3 of 3, 2007.

[24] of Utah, University. Deterlab documentation, Jan. 2000.

[25] Padmanabhan, Venkata N., and Simon, Daniel R. Secure traceroute to detect faulty or malicious routing. *SIGCOMM Computer Communications Review 33* (2002), 77–82.

[26] Perlman, R. *Network layer protocols with byzantine robustness*. PhD thesis, Massachusetts Institute of Technology, 1989.

[27] Publication, Federal Information Processing Standards. Digital signature standard, May 1994.

[28] Thomas, Stephen A. *SSL and TLS Essentials: Securing the Web with CD-ROM.* John Wiley & Sons, Inc., New York, NY, USA, 2000.

[29] Turner, Jonathan S., and Taylor, David E. Diversifying the Internet. In *Proc. of IEEE Global Communications Conference (GLOBECOM)* (Saint Louis, MO, Nov. 2005), vol. 2.

[30] Wendlandt, Dan, and Avramopoulos, Ioannis. Dont secure routing protocols, secure data delivery. In *Proceedings 5th ACM Workshop on Hot Topics in Networks (Hotnets-V* (2006)).

[31] Wikipedia. Sha-1, Dec. 2011.

[32] Wolf, Tilman. In-network services for customization in next-generation networks. *IEEE Network 24*, 4 (July 2010), 6–12.

[33] Yang, Sookhyun, Vasudevan, Sudarshan, and Kurose, Jim. 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.

[34] Zhang, Xin, Jain, Abhishek, and Perrig, Adrian. Packet-dropping adversary identification for data plane security. In *Proceedings of the 2008 ACM CoNEXT Conference* (New York, NY, USA, 2008), CoNEXT '08, ACM, pp. 24:1–24:12.