Detecting Compromised Routers via Packet Forwarding Behavior

Alper Mizrak*, Keith Marzullo, and Stefan Savage

University of California, San Diego

Abstract

While it is widely understood that criminal miscreants are subverting large numbers of Internet-connected computers (e.g., for bots, spyware, SPAM forwarding, etc.) it is less well appreciated that Internet routers are also being actively targeted and compromised. Indeed, due their central role in end-to-end communication, a compromised router can be leveraged to empower a wide range of direct attacks including eavesdropping, man-in-the-middle subterfuge and denial-of-service. In response, a range of specialized anomaly detection protocols has been proposed to detect misbehaving packet forwarding between routers. This paper provides a general framework for understanding the design space of this work and reviews the capabilities of various detection protocols.

Key Words: Internet-wide security, Internet robustness, Fault tolerant routing, Reliable networks, Malicious routers.

1 Introduction

It is widely understood that the Internet is awash in threats. The mean time for a vulnerable system to be infiltrated once connected to the Internet is typically measured in minutes. Consequently, most research effort in network security has focused on protecting these end systems or mitigating the impact of their compromise. However, the Internet architecture relies equally strongly on the correct behavior of the intermediate routers that forward packets, hop-by-hop, to their eventual destination. While it is not as widely appreciated, these network routers are also under attack.

For example, at the 2005 Black Hat Briefings, Mike Lynn demonstrated how Cisco routers can be compromised via simple software vulnerabilities. Even simpler, others have documented how combinations of social engineering and weak passwords can and have been used to compromise thousands of Internet routers and there exists an underground market for trading access to them [1, 2, 3, 4]. Finally, in their annual surveys

*Alper Mizrak is the corresponding author: phone (+1-858-3373791), fax (+1-858-5347029), email (amizrak@cs.ucsd.edu).

of the Internet network security operations community, Arbor Networks Inc., have repeatedly documented such attacks across a range of ISPs [5].

Once a router has been compromised, the standard command line interface from vendors such as Cisco and Juniper can be used to selectively drop packets, delay them or transparently "tunnel" through arbitrary-third party hosts and back again [6, 7]. Together these capabilities are sufficient to selectively eavesdrop, deny or degrade service, or construct a man-in-the-middle attack, against any host that receives service through the router.

Thus, a number of researchers have recently explored the problem of detecting such router compromises as revealed from their inconsistent packet forwarding behavior. In this paper we provide a general framework for understanding this work and then study some of the detection protocols [8, 9, 10, 11] in this context.

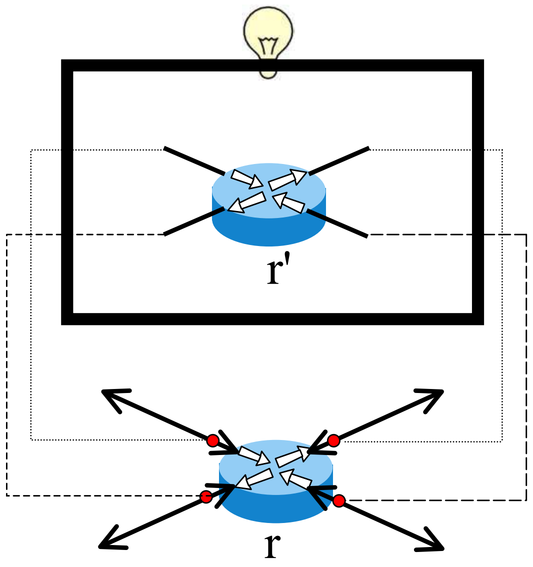

Figure 1. Failure detector via active replica / state machine approach.

2 Centralized Failure Detector via Active Replication

The behavior of a router is deterministic: traffic enters a router and is forwarded on to the next hop towards its destination. Because it is determinsitic, the behavior of a router can be verified by a failure detector via an identical replica of that router.[1] For example, in Figure 1, a failure detector is implemented with an identical replica r' of the router r. In this scheme, the failure detector listens the router r's traffic in promiscuous mode and ensures that the replica r' receives the same input traffic as the router r. Then the failure detector compares the output traffics of the router r and the replica r'. If there is a discrepancy,

[1]This scheme is also called master-checker, active replication, or state machine approach in the literature.

then a failure is detected and an alarm is raised. In this case, either the monitored router is faulty or the failure detector is faulty.

This is an ideal failure detector that detects malicious behavior of compromised routers. However, this scheme has limitations:

Complexity of implementation: First of all, a failure detector must implement the necessary precautions to avoid nondeterminism, such as in scheduling and internal multiplexing. For example, upon receiving routing updates, each router updates its routing tables. If the router and the replica do not update their routing tables simultaneously, then for a short time interval their output traffic might include some discrepancy. Another issue is the randomization used in active queue management schemes. Both the router and the replica rely on the same randomization source to generate the same output.

Researchers addressed this issue by implementing a light-weight version of such a failure detector via *traffic validation*: Instead of validating the exact traffic that transits a router, various characteristics of the traffic entering into and leaving various parts of the network can be used for validation. This is discussed in detail in Section 3.

Resource requirement: Furthermore, to implement such a failure detector, as in Figure 1, requires additional hardware resources — the identical replica of the router — which might be prohibitively expensive.

This limitation is addressed by implementing such a failure detector in a distributed manner. This requires the participation of uncompromised routers. We discuss this issue in Section 4.

3 Traffic Validation

Traffic validation is the basis of detecting anomalous behavior: given traffic entering a region of the network, and knowing the expected behavior of the routers in the network, anomalous behavior is detected when the monitored traffic leaving that part of the network differs significantly from what is expected. Traffic validation can be defined in terms of *conservation of traffic*:

Conservation of traffic: Some *property* of the *traffic* entering into a *region of a network* must be consistent with the same property of the traffic leaving that part of a network.

There are three design decisions that must be addressed to implement such a mechanism:

- Which property of the traffic is to be validated?

  Upon receiving a packet, a router references its routing table to determine the next hop toward the destination, and then forwards the packet. Thus, ideally, the traffic entering into a router is equal to the traffic leaving that router. Of course, there is queueing and processing delay, and packets can be lost due to congestion.

The most precise description of traffic is itself: the exact content of the packets. However, the storage requirements to buffer all packets (as well as the bandwidth consumed by resending them to implement distributed detection, discussed later) make this approach impractical. Instead, one can choose less precise properties of the traffic to validate traffic. Some properties are:

1. Conservation of *flow* validates the volume of the traffic, thereby addressing the malicious behavior of dropping packets.
2. Conservation of *content* validates the content of the traffic, thereyby addressing the malicious behavior of modifying packets.
3. Conservation of *order* validates the order among the packets that constitute the traffic, thereby addressing the malicious behavior of reordering packets.
4. Conservation of *timeliness* validates the time behavior of the forwarding process, thereby addressing the malicious behavior of delaying packets.

- What traffic is to be monitored?

  Some protocols monitor single packet while others monitor aggregate traffic. Some protocols are based on *active probing*: they send probe packets periodically; while others deploy a passive approach that simply monitors existing traffic.

- What region of a network is to be monitored?

  Various existing protocols apply conservation of traffic for different parts of a network, such as per router, per interface, and per path-segment.

A failure detector based on traffic validation can be as effective as one based on active replication, and the overhead is practically feasible. In practice, designing a traffic validation mechanism includes tradeoffs for each design decision above. Hence, implementing a traffic validation mechanism is an engineering problem. In addition, real networks occasionally lose packets due to congestion. Traffic validation needs to accommodate congestive packet loss.

4 Distributed Detection

Instead of a centralized failure detector, as in Figure 1, our goal is to implement such a failure detector distributed in the network using the existing hardware resources: requiring the participation of uncompromised routers.

A compromised router can make arbitrary alterations to the forwarding behavior of that router. However, given the distributed nature of packet forwarding it is not possible in general for an adversary to perfectly conceal such behavior. As long as the packets traverse some uncompromised router, there is enough data redundancy to detect the alteration. The goal is to implement such a failure detector, as in Figure 1, distributed in the network: the detection of a compromised router requires synchronizing the collection of traffic information and distributing the results for detection purposes.

A compromised router can be *traffic faulty* by forwarding traffic in a faulty manner, as well as be *protocol faulty* by behaving arbitrarily with respect to the detection protocol. Thus, any detection protocol should consider the case of routers being protocol faulty and providing bogus traffic information. Attackers can compromise one or more routers in a network. However, we assume that between any two uncompromised routers that there is sufficient path diversity that the malicious routers do not partition the network. In some sense, this assumption is pedantic since it is impossible to guarantee any network communication across such a partition. It is worth noting however, if the host's access router is compromised, then the host is partitioned and there is no routing remedy even if an anomaly is detected; the fate of individual hosts and their access routers are intertwined. In these situations, because of fate-sharing reasons, there is little that can be done. Moreover, from the standpoint of the network such traffic originates from a compromised router, and, therefore, cannot demonstrate anomalous forwarding behavior. To summarize, these distributed detection protocols are designed to detect anomalies between pairs of correct nodes.

As the failure detector via active replica, in Figure 1, has some uncertainty — in case of a detection, either the monitored router is faulty or the failure detector is faulty — there will be some uncertainty, as well, in distributed detection determining which router is faulty, since routers collect the information upon which traffic validation is based. For example, suppose router r_1 collects traffic information about packets that traverse r_1, then a neighboring router r_2, and then a third router r_3. Based on the information r_3 has about the traffic it has seen and the traffic information r_1 has provided, r_3 can determine that packets has been dropped. But, r_3 can't determine whether r_1 is lying about what it claims to have forwarded to r_2 or whether r_2 has dropped the packets. Hence, there is an inherent lack of precision in determining which routers are compromised.

A failure detector reports a suspicion as a *path-segment*, which is defined as a sequence of consecutive routers that is a subsequence of a path.[2] More specifically, a failure detector reports a path-segment if it suspects a router in that path-segment behaving in a faulty manner.

[2]For example, if a network consists of the single path $\langle r_1, r_2, r_3, r_4\rangle$ then $\langle r_2, r_3\rangle$ is a path segment, but $\langle r_1, r_3\rangle$ is not because r_1 and r_3 are not adjacent.

This section is a brief summary of the formal specification presented in [12]. We refer the readers to [12] for the derivation of the formal specification. In short, we cast the problem as a failure detector with *completeness*, *accuracy* and *precision* properties.

Completeness: Whenever a router forwards traffic in a faulty manner,

- if *all correct routers* eventually suspect a path-segment containing a faulty router, then a failure detector is *strong-complete.*
- if *at least one correct router* eventually suspects a path-segment containing a faulty router, then a failure detector is *weak-complete.*

Accuracy: A failure detector is *accurate* if, whenever a correct router suspects a path-segment, then there is at least one faulty router in that path-segment.

Precision: A failure detector also has a *precision*, which is the maximum length of a path-segment it suspects.

A failure detector must be complete and accurate, and preferably with a *high precision.* Implementing such distributed detection involves tradeoffs among precision, weak/strong-completeness, and the overhead of monitoring and communication. Various detection protocols address these design decisions in different ways, which we study in Section 5.

Compared to weak-completeness, strong-completeness is more desirable property since every correct router detects the fault. Given a weak-complete detector, a strong-complete one can be implemented but it may not be simple and some precision would be lost. For example, consider that a source router r_s detects a link $\langle r_1, r_2 \rangle$ as faulty. Announcing this detection, the other correct routers in the network have to consider the case that r_s being faulty as well. On the other hand, in some cases, having a weak-complete detector is enough for taking proper response: for example, relying on source routing, the router r_s may only update its own routing table excluding the suspected $\langle r_1, r_2 \rangle$.

5 Case Studies

In this section, we study various protocols proposed as a countermeasure for the attacks on the network data plane. First, we present each traffic validator that is implemented by the protocols as a single centralized service, and examine their various design decisions. Next, we study how the existing protocols have implemented such failure detectors by distributing them in the network.

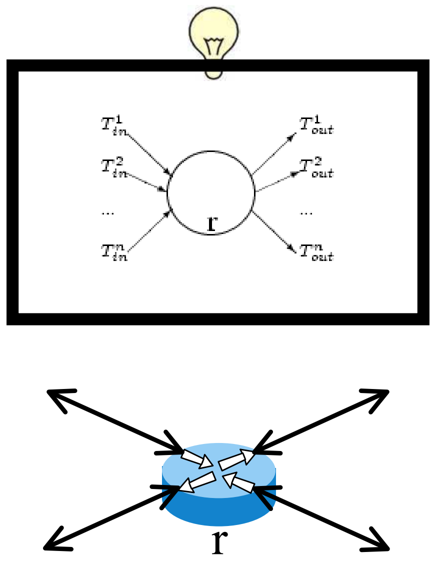

Figure 2. Failure detector via a traffic validator per router.

5.1 Traffic Validation Per Router: `WATCHERS`

The most similar approximation to the failure detector in Figure 1 is `WATCHERS` [8], which detects and isolates faulty routers based on a distributed network monitoring approach. A faulty router is defined as one that drops or misroutes packets, or that behaves in an arbitrary manner with respect to the proposed protocol. A *conservation of flow principle* (CoFP) was proposed to detect faulty routers. Basically, CoFP states that the amount of traffic entering into a router should be equal to the amount of traffic leaving that router.

The traffic validator that `WATCHERS` implements is given in Figure 2 as a centralized service. `WATCHERS` validates the conservation of flow property of the aggregate traffic entering into each router in the network. If the difference between the volumes of the traffic entering into and leaving the router exceeds a user-defined threshold, then a failure is detected and an alarm is raised. This threshold is needed to avoid false positives as a result of congestive packet losses.

`WATCHERS` implements this failure detector by requiring all the neighboring routers of a router r to synchronize with each other, to count how many bytes it has received from and forwarded to r during an agreed-upon time interval, to distribute the snapshots of its counters to the others by flooding, and finally to validate the conservation of flow property.

If a neighbor router r_n can not validate the router r, then r_n announces that the link $\langle r_n, r\rangle$ is suspicious and $\langle r_n, r\rangle$ is removed from the routing fabric.

In terms of our specification, it is *accurate* with a *precision* of 2. `WATCHERS` is not complete since it has a flaw as explained in [13]. It can also be fixed [13], in which case WATCHERS is strong-complete.

Generally speaking, there are two limitations of WATCHERS:

- The main drawback of WATCHERS is its weak traffic validation, which is designed for a restrictive threat model: it addresses only malicious packet drops and misroutes. Several researchers have subsequently developed protocols with more general traffic validation mechanisms addressing a comprehensive set of attacks.

- The architects of WATCHERS noticed a completeness problem caused by consorting faulty routers, which is first introduced by Perlman [14], the earliest work on fault-tolerant forwarding. Consorting faulty router are the faulty routers launching a coordinated attack and cooperating to hide each others malicious behavior. WATCHERS addressed the issue by requiring each router to maintain a separate state for every neighbor and destination pair in the network. Other protocols addressed this problem of consorting faulty routers with a different approach: they validate traffic over path-segments. We discuss this next.

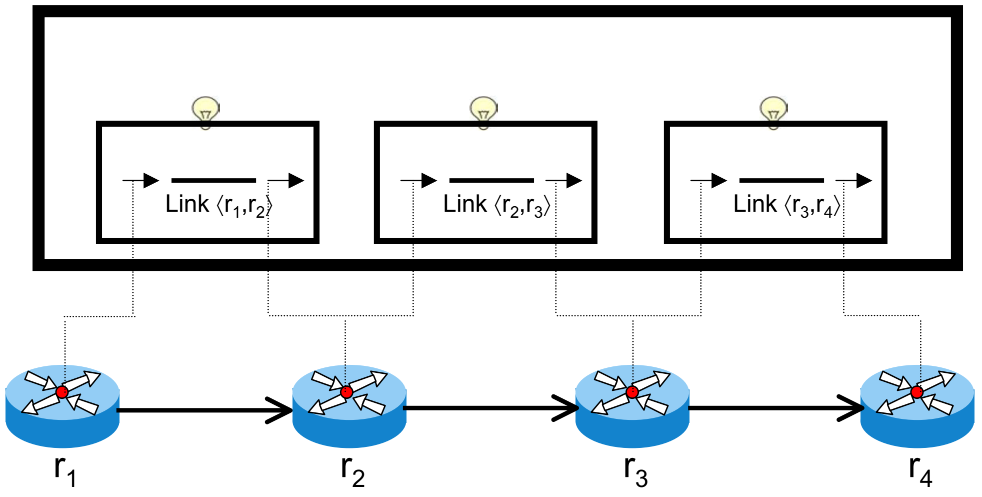
Link $\langle r_1,r_2 \rangle$
Link $\langle r_2,r_3 \rangle$
Link $\langle r_3,r_4 \rangle$
r_1
r_2
r_3
r_4

Figure 3. Failure detector via a traffic validator per path-segment noeds.

5.2 Traffic Validation Per Path-segment Nodes: HSER

[9] presents Highly Secure and Efficient Routing (HSER), a combination of source routing, hop by hop authentication, a-priori reserved buffers, sequence numbers, timeouts, end-to-end reliability mechanisms, and fault announcements. As it is noted in [9], while none of these individual mechanisms is novel by itself, it is the combination of them that delivers Byzantine robustness and detection.

The traffic validator that `HSER` implements is given in Figure 3 as a centralized service. `HSER` validates the conservation of content property of a single packet that is monitored along the path from the source to the destination. If any router along the path discovers that its neighbor has lost or altered the packet, then a failure is detected and an alarm is raised.

`HSER` implements such a failure detector distributed in the network by requiring each router along the path to compute a fingerprint for the monitored packet, and to keep a timeout, and finally to validate the conservation of content property with its neighbors. Upon receiving a packet, the router first validates the authenticity and forwards the packet to the next hop towards the destination. After forwarding the packet, the router sets a timeout to the worst case round trip time to the destination from itself. If authenticity of the packet is not verified or the timeout expires, then the router generates a *fault announcement*, including its neighbor and itself, to send back to the source.

`HSER` relies on source routing. As a response, upon receiving a fault announcement, the source router computes a new route to the destination excluding the detected link from its routing fabric.

In terms of our specification, `HSER` is *weak-complete* – since only the source detects a failure – and *accurate* with a *precision* of 2.

The overhead of this approach is prohibitively high, since for every source and destination pair, all of the routers along the path have to participate in the detection. There are two tradeoffs that the researchers decided to implement a feasible detector for practical deployment based on traffic validation per path-segment ends:

- Give up precision by only validating at the end routers of the path-segment, in which case none of the intermediate routers along the path participates in detection.
- The end routers of the path-segment can decide on a sampling pattern and keep track of only the chosen packets. This method may help decreasing overhead significantly. However, the attacks to those unmonitored packets would not lead to a detection, and the accuracy in detection would be decreased.

Other protocols based on this approach are presented in [12, 15, 16].

5.3 Traffic Validation Per Path-segment Ends: `SecTrace`

`SecTrace` [10] is developed as a practical tool to securely trace the path of an existing traffic towards a particular destination from a source. It proceeds hop-by-hop similar to Traceroute: at each round, the source validates the traffic between itself and an intermediate router towards the destination.

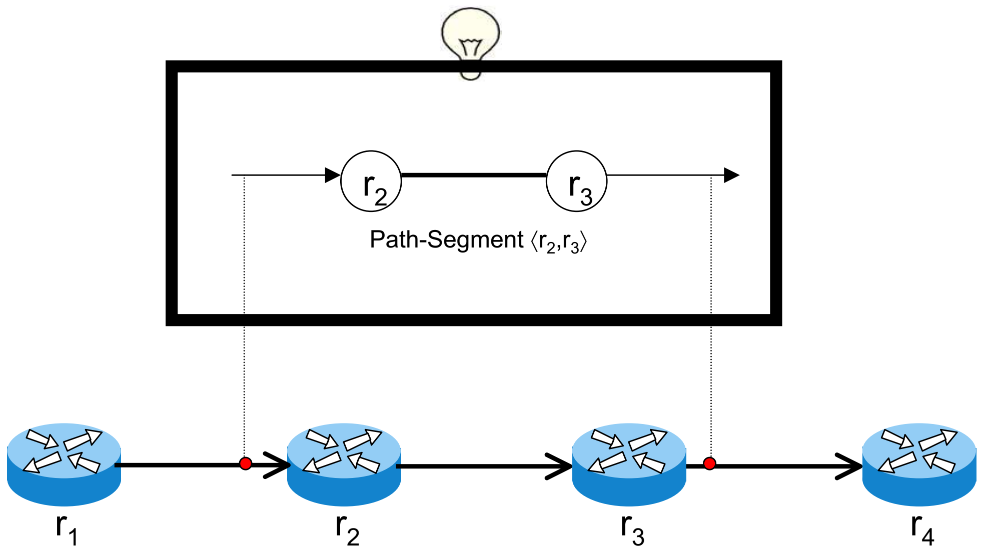

Figure 4. Failure detector via a traffic validator per path-segment ends.

The traffic validator that `SecTrace` implements is given in Figure 4 as a centralized service. `SecTrace` validates the conservation of content property of the aggregate or sampled traffic[3] between the source router and an intermediate router. If the source detects that there is discrepancy in the traffic, then a failure is detected and an alarm is raised.

`SecTrace` implements such a failure detector distributed in the network by requiring only the end routers of the monitored path-segment to synchronize with each other and to compute fingerprints for the traffic between themselves for an agreed-upon time interval. At the end of the round, the corresponding intermediate router sends back the information it has collected and the identity of the next expected router towards the destination. Upon receiving this information, the source router validates the conservation of content property: if the source validates the traffic, then it initiates another `SecTrace` round with the next intermediate router towards the destination; otherwise, the source detects a failure.

For example, in Figure 4, a path-segment of $\langle r_1, r_2, r_3, r_4 \rangle$ is monitored during the given traffic validation round and only the source r_1 and the corresponding intermediate router r_4 implement the distributed failure detector. If the source r_1 detects that there is discrepancy in the traffic, then a failure is detected and an alarm is raised: (1) Either one of the intermediate routers $\{r_2, r_3\}$ is traffic faulty introducing discrepancy into the monitored traffic. (2) Or, the failure detector, which is implemented by r_1 and r_4, is protocol faulty: at least one of $\{r_1, r_4\}$ is faulty.

In terms of our specification, `SecTrace` is *weak-complete* – since only the source detects a failure – and *accurate* with a *precision* of k, where k is the length of the monitored path-segment.

[3]It can adopt both active probing and passive monitoring approaches.

On the other hand, in [10], the authors require that the source detects the link between the corresponding intermediate router and its upstream neighbor. For example, in Figure 4, if the source r_1 could not validate the traffic with r_4, then it would detect $\langle r_3, r_4 \rangle$ as faulty. The reasoning behind this approach is that the source r_1 was able to validate the same traffic up to the upstream neighbor r_3 at the previous validation round, so either r_3 or r_4 must be faulty introducing discrepancy into the traffic. However, this approach violates the accuracy property. Assume that only the router r_2 is faulty manipulating the traffic only after the source r_1 validates the traffic with r_3. Consequently, $\langle r_3, r_4 \rangle$ would be detected where neither r_3 nor r_4 is faulty. To address this problem, the authors propose to give occasional indications of `SecTrace` activity, such as by continuously sending round initialization packets pretending to monitor the traffic, while in reality doing nothing.

Finally, they propose three different approaches as a response: (1) The source tries to route the traffic around the detected link using source routing. (2) The source notifies the downstream routers, expecting them to make the appropriate routing adjustments avoiding the suspected routers. (3) The source alerts the administrator of the suspected routers.

Other protocols based on this approach are presented in [17, 12, 18, 19, 20, 21].

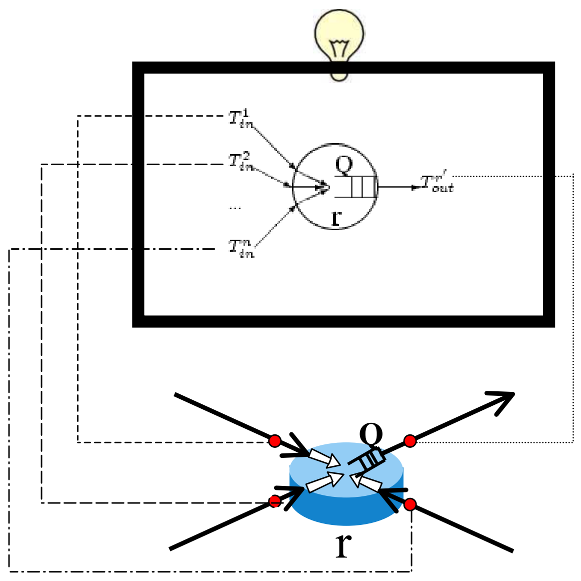

Figure 5. Failure detector via a traffic validator per interface.

5.4 Traffic Validation Per Interface: `Protocol` χ

Unfortunately, it is quite challenging to attribute a missing packet to a malicious action because modern networks routinely drop packets when the load temporarily exceeds a router's buffering capacity. Almost all detection protocols have tried to address this problem using a user-defined threshold. Using such a threshold will necessarily create unnecessary false positives or mask highly-focused attacks. `Protocol` χ, which is a compromised router detection protocol recently presented in [11], dynamically infers the number of congestive packet losses that will occur based on measured traffic rates and buffer sizes. Once the ambiguity from congestion is removed, subsequent packet losses can be attributed to malicious actions.

The traffic validator that `protocol` χ implements is given in Figure 5 as a centralized service. `Protocol` χ validates the conservation of timeliness property of the aggregate traffic entering into each interface of a router in the network. If a packet loss occurs when the monitored interface's queue is not full, then a failure is detected and an alarm is raised.

`Protocol` χ implements such a failure detector distributed in the network by requiring each neighbor of a router to synchronize with each other, to compute a fingerprint with a timestamp for each packet passes through the interface Q during an agreed-upon time interval, to distribute these information among the other neighboring routers, and finally to validate the conservation of timeliness property of the traffic by simulating the behavior of the monitored interface queue.

If a neighbor router r_n detects that the router r drops a packet when the corresponding queue is not full, then r_n announces that the link $\langle r_n, r\rangle$ is suspicious and $\langle r_n, r\rangle$ is removed from the routing fabric.

In terms of our specification, `protocol` χ is *strong-complete* and *accurate* with a *precision* of 2.

`Protocol` χ can be extended addressing the adjacent consorting faulty routers by monitoring every output interfaces of the neighbors k hops away and disseminating the traffic information to all neighbors within diameter hops. This is the same approach that was used in [12], and it increases the overhead of detection.

[22] is also based on this approach.

6 Conclusion

In this paper we have described a general framework for understanding the literature on detecting malicious routers via packet forwarding behavior. We have described how traffic validation is the basis for all such schemes and distributed per-router validation is a simple approximation to an idealized detector. However, due to the threat of consorting faulty routers (i.e., that multiple routers have been compromised) practical systems are limited to validating path segments rather than individual routers. Within this approach there is a tradeoff between precision and overhead – depending on the span of a segment and most protocols have chosen to explore the lower-overhead part of this design space. Finally, all protocols are subject to noise due to congestive packet loss which is difficult to distinguish from malicious dropping. While per-interface techniques can differentiate between these conditions, it comes at the expense of high overhead. In a short time, there have been significant advances in this domain, and while none of these protocols has yet been deployed in a production network, they are quickly becoming cheap enough and precise enough to be a viable option against router-oriented attacks.

References

[1] R. Thomas, "ISP Security BOF, NANOG 28," Jun. 2003, http://www.nanog.org/mtg-0306/pdf/thomas.pdf.

[2] X. Ao, "DIMACS Report: Workshop on Large Scale Internet Attacks," Nov. 2003.

[3] M. Wolfgang, "Exploiting Cisco Routers," Sep. 2003, http://www.securityfocus.com/infocus/1734.

[4] M. Aharoni and W. M. Hidalgo, "Cisco SNMP configuration attack with a GRE tunnel," Sep. 2005, http://www.securityfocus.com/infocus/1847.

[5] D. McPherson and C. Labovitz, "Worldwide Infrastructure Security Report," Sep. 2006, http://www.arbornetworks.com/sp_security_report.php.

[6] Gauis, "Things to do in Ciscoland when you're dead," Jan. 2000, www.phrack.org.

[7] D. Taylor, "Using a compromised router to capture network traffic," Jul. 2002, unpublished Technical Report.

[8] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson, "Detecting disruptive routers: A distributed network monitoring approach," in *Proceedings of the IEEE Symposium on Security and Privacy*, May 1998, pp. 115–124.

[9] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy, "Highly secure and efficient routing," in *Proceedings of INFOCOM 2004 Conference*, March 2004, — "Amendment to: Highly Secure and Efficient Routing; Feb 2004".

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

[11] A. T. Mizrak, K. Marzullo, and S. Savage, "Detecting malicious packet losses," UCSD, Tech. Rep. CS2007-0889, April 2007.

[12] 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.

[13] A. T. Mizrak, K. Marzullo, and S. Savage, "Detecting malicious routers," UCSD, Tech. Rep. CS2004-0789, MAY 2004.

[14] R. Perlman, "Network layer protocols with byzantine robustness," Ph.D. dissertation, MIT LCS TR-429, Oct. 1988.

[15] K. Argyraki, P. Maniatis, D. Cheriton, and S. Shenker, "Providing packet obituaries," in *Proceedings of ACM SIGCOMM HotNets-III*, 2004.

[16] A. Herzberg and S. Kutten, "Early detection of message forwarding faults," *SIAM J. Comput.*, vol. 30, no. 4, pp. 1169–1196, 2000.

[17] C. N.-R. Baruch Awerbuch, David Holmer and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," in *ACM Workshop on Wireless Security (WiSe)*, September 2002.

[18] I. Avramopoulos and J. Rexford, "Stealth Probing: Efficient Data-Plane Security for IP Routing," in *Proc. USENIX Annual Technical Conference*, May-Jun 2006.

[19] S. Lee, T. Wong, and H. S. Kim, "Secure split assignment trajectory sampling: A malicious router detection system," *dsn*, vol. 0, pp. 333–342, 2006.

[20] D. Wendlandt, I. Avramopoulos, D. Andersen, and J. Rexford, "Don't Secure Routing Protocols, Secure Data Delivery," in *Proc. 5th ACM Workshop on Hot Topics in Networks (Hotnets-V)*, Irvine, CA, Nov. 2006.

[21] S. Goldberg, D. Xiao, B. Barak, and J. Rexford, "Measuring path quality in the presence of adversaries: The role of cryptography in network accountability," Princeton University, Tech. Rep., 2007.

[22] W. Zhang, R. Rao, G. Cao, and G. Kesidis, "Secure routing in ad hoc networks and a related intrusion detection problem," in *IEEE Military Communications Conference*, 2003.