International Journal of Computer Science and Informatics

Volume 2 | Issue 4 Article 3

April 2013

Implementation of RED Algorithm to Detect Malicious Packet Losses

G. Sanjay Gandhi
VISIT Engineering College ,Tadepalli Gudem, Sanjaygandhi.g@gmail.com

S. S. V. Appa Rao
VISIT Engineering College ,Tadepalli Gudem, surisettysivaji@gmail.com

V.V.V. Satyanarayana Reddy
S. D. College of Information Tech, Tanuku, v_velagala@yahoo.com

Follow this and additional works at: https://www.interscience.in/ijcsi

 Part of the Computer Engineering Commons, Information Security Commons, and the Systems and Communications Commons

Recommended Citation

Gandhi, G. Sanjay; Rao, S. S. V. Appa; and Reddy, V.V.V. Satyanarayana (2013) "Implementation of RED Algorithm to Detect Malicious Packet Losses," *International Journal of Computer Science and Informatics*: Vol. 2 : Iss. 4 , Article 3.
Available at: https://www.interscience.in/ijcsi/vol2/iss4/3

This Article is brought to you for free and open access by Interscience Research Network. It has been accepted for inclusion in International Journal of Computer Science and Informatics by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

Implementation of RED Algorithm to Detect Malicious Packet Losses

G. Sanjay Gandhi[1] , S. S. V. Appa Rao[2] & V.V.V. Satyanarayana Reddy[3]

[1&2]VISIT Engineering College ,Tadepalli Gudem, [3]S. D. College of Information Tech, Tanuku
E-mail : Sanjaygandhi.g@gmail.com[1], surisettysivaji@gmail.com[2], v_velagala@yahoo.com[3]

Abstract - We consider the problem of detecting whether a compromised router is maliciously manipulating its stream of packets. In particular, we are concerned with a simple yet effective attack in which a router selectively drops packets destined for some victim. Unfortunately, it is quite challenging to attribute a missing packet to a malicious action because normal network congestion can produce the same effect. Modern networks routinely drop packets when the load temporarily exceeds their buffering capacities. Previous detection protocols have tried to address this problem with a user -defined threshold: too many dropped packets imply malicious intent. However, this heuristic is fundamentally unsound; setting this threshold is, at best, an art and will certainly create unnecessary false positives or mask highly focused attacks. We have designed, developed, and implemented a compromised router detection protocol that dynamically infers, based on measured traffic rates and buffer sizes, the number of congestive packet losses that will occur. Once the ambiguity from congestion is removed, subsequent packet losses can be attributed to malicious actions.

Keywords - *Internet dependability, intrusion detection and tolerance, distributed systems, reliable networks, malicious routers.*

I. INTRODUCTION

The Internet is not a safe place. Unsecured hosts can expect to be compromised within minutes of connecting to the Internet and even well-protected hosts may be crippled with denial-of-service (DoS) attacks. However, while such threats to host systems are widely understood, it is less well appreciated that the network infrastructure itself is subject to constant attack as well. Indeed, through combinations of social engineering and weak passwords, attackers have seized control over thousands of Internet routers [1], [2]. Even more troubling is Mike Lynn's controversial presentation at the 2005 Black Hat Briefings, which demonstrated how Cisco routers can be compromised via simple software vulnerabilities. Once a router has been compromised in such a fashion, an attacker may interpose on the traffic stream and manipulate it maliciously to attack others—selectively roping, modifying, or rerouting packets.

Several researchers have developed distributed protocols to detect such traffic manipulations, typically by Validating that traffic transmitted by one router is received unmodified by another [3], [4]. However, all of these schemes—including our own—struggle in interpreting the absence of traffic. While a packet that has been modified in transit represents clear evidence of tampering, a missing packet is inherently ambiguous: it may have been explicitly blocked by a compromised router or it may have been dropped benignly due to network congestion. In fact, modern routers routinely drop packets due to bursts in traffic that exceed their buffering capacities, and the widely used Transmission Control Protocol (TCP) is designed to cause such losses as part of its normal congestion control behavior. Thus, existing traffic validation systems must inevitably produce false positives for benign events and/or produce false negatives by failing to report real malicious packet dropping.

We develop a compromised router detection protocol that dynamically infers the precise number of congestive packet losses that will occur. Once the congestion ambiguity is removed, subsequent packet losses can be safely attributed to malicious actions. We believe our protocol is the first to automatically predict congestion in a systematic manner and that it is necessary for making any such network fault detection practical.

We have evaluated our protocol in a small experimental network and demonstrate that it is capable of accurately resolving extremely small and fine-grained attacks.

II. EXISTING SYSTEM:

In building a traffic validation protocol, it is necessary to explicitly resolve the ambiguity around

packet losses. Should the absence of a given packet be seen as malicious or benign? In practice, there are three approaches for Addressing this issue: . Static Threshold. Low rates of packet loss are assumed to be congestive, while rates above some predefined threshold are deemed malicious. . Traffic modeling. Packet loss rates are predicted as a function of traffic parameters and losses beyond the prediction are deemed malicious. . Traffic measurement. Individual packet losses are predicted as a function of measured traffic load and router buffer capacity. Deviations from these Predictions are deemed malicious. Most traffic validation protocols, including WATCHERS [3], Secure Trace route [12], and our own work described in [4], analyze aggregate traffic over some period of time in order to amortize monitoring overhead over many packets. For example, one validation protocol described in [4] maintains packet counters in each router to detect if traffic flow is not conserved from source to destination. When a packet arrives at router r and is forwarded to a destination that will traverse a path segment ending at router x, r increments an outbound counter associated with router x. Conversely, when a packet arrives at router r, via a path segment beginning with router x, it increments its inbound counter associated with router x. periodically, router x sends a copy of its outbound counters to the associated routers for validation.

Then, a given router r can compare the number of packets that x claims to have sent to r with the number of packets it counts as being received from x, and it can detect the number of packet losses. Thus, over some time window, a router simply knows that out of m packets sent, n were successfully received. To address congestion ambiguity, all of these systems employ a predefined threshold: if more than this number is dropped in a time interval, then one assumes that some router is compromised. However, this heuristic is fundamentally flawed: how does one choose the threshold? In order to avoid false positives, the threshold must be large enough to include the maximum number of possible congestive legitimate packet losses over a measurement interval. Thus, any compromised router can drop that many packets without being detected. Unfortunately, given the nature of the dominant TCP, even small numbers of losses can have significant impacts. Subtle attackers can selectively target the traffic flows of a single victim and within these flows only drop those packets that cause the most harm. For example, losing a TCP SYN packet used in connection establishment has a disproportionate impact on a host because the retransmission time-out must necessarily be very long (typically 3 seconds or more). Other seemingly minor attacks that cause TCP time-outs can have similar effects—a class of attacks well described.

All things considered, it is clear that the static threshold mechanism is inadequate since it allows an attacker to mount vigorous attacks without being detected. Instead of using a static threshold, if the probability of congestive losses can be modeled, then one could resolve ambiguities by comparing measured loss rates to the rates predicted by the model. One approach for doing this is to predict congestion analytically as a function of individual traffic flow parameters, since TCP explicitly responds to congestion. Indeed, the behavior of TCP has been excessively studied.

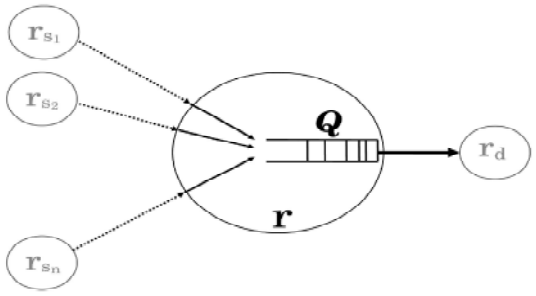


Fig 1: Validating the queue of an output interface.

A router can be traffic faulty by maliciously dropping packets and protocol faulty by not following the rules of the detection protocol. We say that a compromised router r is traffic faulty with respect to a path segment _ during _ if _contains r and, during the period of time _, r maliciously drops or misroutes packets that flow through _. A router can drop packets without being faulty, as long as the packets are dropped because the corresponding output interface is congested. A compromised router r can also behave in an arbitrarily malicious way in terms of executing

The protocol we present, in which case we indicate r as protocol faulty. A protocol faulty router can send control Messages with arbitrarily faulty information or it can simply not send some or all of them. A faulty router is one that is traffic faulty, protocol faulty, or both. Attackers can compromise one or more routers in a network. However, for simplicity, we assume in this paper that adjacent routers cannot be faulty. Our work is easily extended to the case of k adjacent faulty routers.

Disadvantages of Existing System :

- Thus, existing traffic validation systems must inevitably produce false positives for benign events and/or produce false negatives by failing to report real malicious packet dropping.

- Previous work has approached this issue using a static user-defined threshold, which is fundamentally limiting.

Proposed Work :

Using a Compromised Router to Capture Network Traffic Using Red Algorithm.

In this paper we Propose the we are designed, developed, and implemented a compromised router detection protocol that dynamically infers, based on measured traffic rates and buffer sizes, the number of congestive packet losses that will occur. Once the ambiguity from congestion is removed, subsequent packet losses can be attributed to malicious actions.

The approach chosen was to establish a GRE tunnel between the captured router ("Target router") and a second router that is under the control of the attacker ("Attacker router"). Policy routing was then used to redirect ingress and egress traffic for the organization to the attacker router via the GRE tunnel. The traffic was then 'handled' by the attacker router before being returned to the target router for final delivery (again via the GRE tunnel).

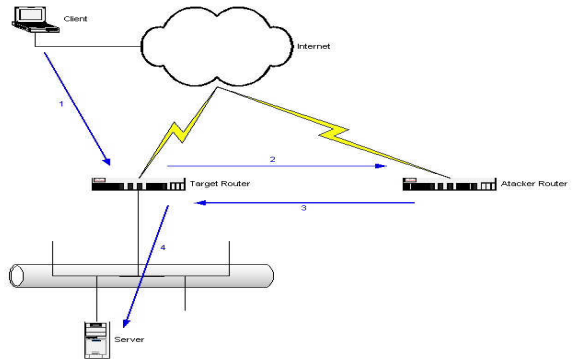


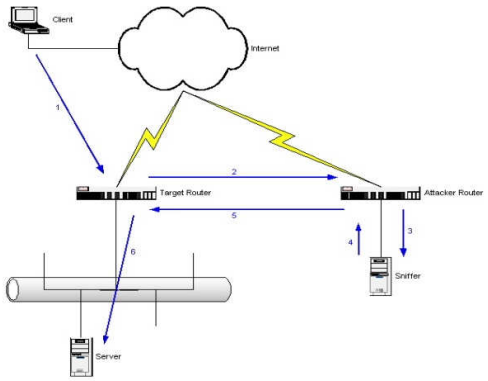
Fig. 2

Two handling scenarios were tested. In the first, the captured traffic was merely 'reflected' by the attacker router back down the GRE tunnel; This method had the advantage of simplicity in the router configuration, but introduced the following issues:

In order to capture the traffic it is necessary to 'sniff' the external interface of the attacker router. This would be somewhat difficult for non-Ethernet network media.

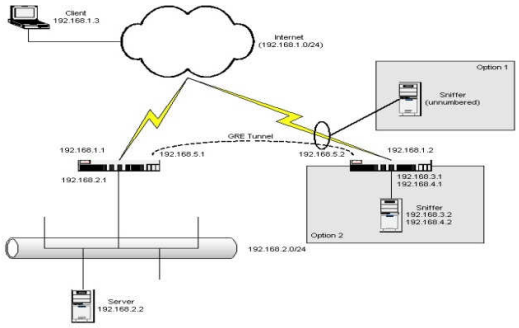
Captured network traffic is GRE encapsulated. It would be necessary to encapsulate this traffic before an IP decode could be performed.

In the second handling scenario, the attacker router was configured to pass the captured traffic by a Unix workstation before sending it back to the target router. This is shown in Figure 2. This scenario overcomes the two previous disadvantages:



- The external network media on the attacker router is arbitrary.
- The traffic forwarded via the Unix workstation has already been de capsulated, and requires less processing to extract useful information

The earliest work on fault-tolerant forwarding is due to Perlman [11] who developed a robust routing system based on source routing, digitally signed route-setup packets, and reserved buffers.



While groundbreaking, Perlman's work required significant commitments of router resources and high levels of network participation to detect anomalies. Since then, a variety of researchers have proposed lighter weight protocols for actively probing the network to test whether packets are forwarded in a manner consistent with the advertised global topology [5], [11], [12]. Conversely, the 1997 WATCHERS system detects disruptive routers passively via a distributed monitoring algorithm that detects deviations from a "conservation of flow" invariant [12], [3]. However, work on WATCHERS was abandoned, in part due to limitations

in its distributed detection protocol, its overhead, and the problem of ambiguity stemming from congestion [12]. Finally, our own work broke the problem into three pieces: a traffic validation mechanism, a distributed detection protocol, and a rerouting countermeasure. In [11] and [4], we focused on the detection protocol, provided a formal framework for evaluating the accuracy and precision of any such protocol, and described several practical protocols that allow scalable implementations. However, we also assumed that the problem of congestion ambiguity could be solved, without providing a solution. This paper presents a protocol that removes this assumption.

III. ANALYSIS OF PROTOCOL

We cast the problem of detecting compromised routers as a failure detector with accuracy and completeness properties. There are two steps in showing the accuracy and completeness of _: Showing that TV is correct. . Showing that _ is accurate and complete assuming that TV is correct. Assuming that there exists no adjacent faulty routers, we show in Appendices B and C that if TV is correct, then _ is 2-accurate and 2-complete, where 2 indicates the length of detection: A link consisting of two routers is detected as a result.

Any failure of detecting malicious attack by TV results in a false negative, and any misdetection of legitimate behavior by TV results in a false positive. Within the given system model of Section 4, the example TV predicate is correct. However, the system model is still simplistic. In a real router, packets may be legitimately dropped due to reasons other than congestion: for example, errors in hardware, software or memory, and transient link errors. Classifying these as arising from a router being compromised might be a problem, especially if they are infrequent enough that they would be best ignored rather than warranting repairs the router or link. A larger concern is the simple way that a router is modeled in how it internally multiplexes packets. This model is used to compute time stamps. If the time stamps are incorrect, then TV could decide incorrectly. We hypothesize that a sufficiently accurate timing model of a router is attainable but have yet to show this to be the case. A third concern is with clock synchronization. This version of TV requires that all the routers feeding a queue have synchronized clocks. This requirement is needed in order to ensure that the packets are interleaved correctly by the model of the router. The synchronization requirement is not necessarily daunting; the tight synchronization is only required by routers adjacent to the same router. With low-level time stamping of packets and repeated exchanges of time, it should be straightforward to synchronize the clocks sufficiently tightly. Other representations of collected traffic information and TV that we have considered have their own problems with false positives and false negatives. It is an open question as to the best way to represent TV. We suspect any representation will admit some false positives or false negatives.

The main overhead of protocol _ is in computing a fingerprint for each packet. This computation must be done at wire speed. Such a speed has been demonstrated to be attainable. In our prototype, we implemented fingerprinting using UHASH [28]. Rogaway [29] demonstrated UHASH performance of more than 1 Gbps on a 700-MHz Pentium III processor when computing a 4-byte hash value. This performance could be increased further with hardware support. Network processors are designed to perform highly parallel actions on data packets [30]. For example, Feghali et al. [31] presented an implementation of wellknown private-key encryption algorithms on the Intel IXP28xx network processors to keep pace with a 10-Gbps forwarding rate. Furthermore, Sanchez et al. [32] demonstrated hardware support to compute fingerprints at wire speed of high speed routers (OC-48 and faster).

Advantage :

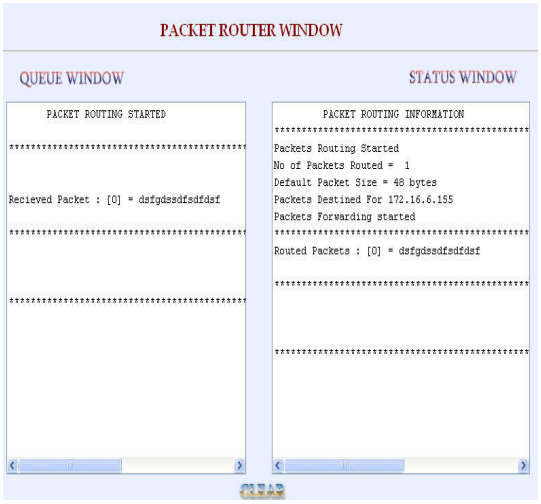
Finally, our own work broke the problem into three pieces:

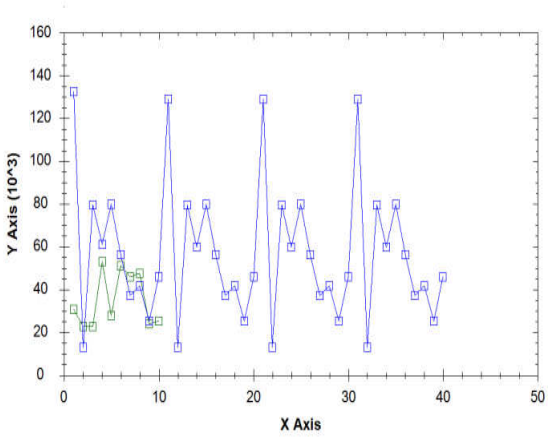
1. A traffic validation mechanism
2. A distributed detection protocol
3. And a rerouting countermeasure.

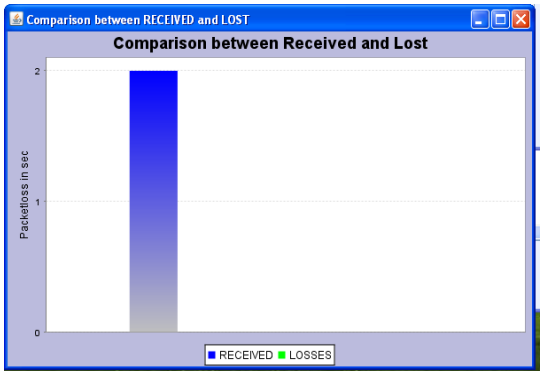
IV. EXPERIMENTAL RESULTS

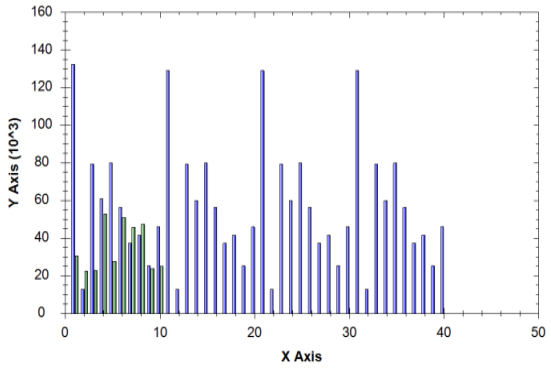
We have implemented and experimented with protocol _ in the Emulab tested. In our experiments, we used the simple topology shown in Fig. 3. The routers were Dell Power Edge 2850 PC nodes with a single 3.0-GHz 64-bit Xeon processor and 2 Gbytes of RAM, and they were running Redhat-Linux-9.0 OS software. Each router except for r1 was connected to three LANs to which user machines were connected. The links between routers were configured with 3-Mbps bandwidth, 20-ms delay, and 75,000-byte capacity FIFO queue. Each pair of routers shares secret keys; furthermore, integrity and authenticity against the message tampering is provided by message authentication codes. The validation time interval _ was set to 1 second, and the upper bound on the time to forward traffic information _ was set to 300 ms. At the end of each second, the routers exchanged traffic information corresponding to the last validation interval and evaluated the TV predicate after 2_ ¼ 600 ms. Each run in an experiment consisted of an execution of 80 seconds. During the first 30 seconds, we generated no traffic to allow the routing fabric to initialize. Then, we generated 45 seconds of traffic.

We then experimented with the ability of protocol _ to detect attacks. In these experiments, the router r1 is compromised to attack the traffic selectively in various ways, targeting two chosen ftp flows. The duration of the attack is indicated with a line bounded by diamonds in the figures, and a detection is indicated by a filled circle. For the first attack, the router r1 was instructed to drop 20 percent of the selected flows for 10 seconds. Predicted queue length and the confidence values for each packet drop can be seen in Figs. 5a and 5b. As shown in the graph, during the attack, protocol _ detected the failure successfully. In the second attack, router r1 was instructed to drop packets in the selected flows when the queue was at least 90 percent full. Protocol _ was able to detect the attack and raised alarms









V. CONCLUSIONS AND FUTURE WORK

To the best of our knowledge, this paper is the first serious attempt to distinguish between a router dropping packets maliciously and a router dropping packets due to congestion. Previous work has approached this issue using a static user-defined threshold, which is fundamentally limiting. Using the same framework as our earlier work (which is based on a static user-defined threshold) [4], we developed a compromised router detection protocol _ that dynamically infers, based on measured traffic rates and buffer sizes, the number of congestive packet losses that will occur. Subsequent packet losses can be attributed to malicious actions. Because of no determinism introduced by imperfectly synchronized clocks and scheduling delays, protocol _ uses user-defined significance levels, but these levels are independent of the properties of the traffic. Hence, protocol _ does not suffer from the limitations of static thresholds. We evaluated the effectiveness of protocol _ through an implementation and deployment in a small network. We show that even fine-grained attacks, such as stopping a host from opening a connection by discarding the SYN packet, can be detected.

REFERENCES

[1] X. Ao, Report on DIMACS Workshop on Large-Scale Internet Attacks, http://dimacs.rutgers.edu/ Workshops/Attacks/internet-attack-9-03.pdf, Sept. 2003.

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

[3] K.A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R.A. Olsson, "Detecting Disruptive Routers: A Distributed Network Monitoring Approach," Proc. IEEE Symp. Security and Privacy (S&P '98), pp. 115-124, May 1998.

[4] A.T. Mizrak, Y.-C. Cheng, K. Marzullo, and S. Savage, "Detecting and Isolating Malicious Routers," IEEE Trans. Dependable and Secure Computing, vol. 3, no. 3, pp. 230-244, July-Sept. 2006.

[5] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz, "Listen and Whisper: Security Mechanisms for BGP," Proc. First Symp. Networked Systems Design and Implementation (NSDI '04), Mar. 2004.

[6] S. Kent, C. Lynn, J. Mikkelson, and K. Seo, "Secure Border Gateway Protocol (Secure-BGP)," IEEE J. Selected Areas in Comm., vol. 18, no. 4, pp. 582-592, Apr. 2000.

[7] Y.-C. Hu, A. Perrig, and D.B. Johnson, "Ariadne: A Secure On- Demand Routing Protocol for Ad Hoc Networks," Proc. ACM MobiCom '02, Sept. 2002.

[8] B.R. Smith and J. Garcia-Luna-Aceves, "Securing the Border Gateway Routing Protocol," Proc. IEEE Global Internet, Nov. 1996.

[9] S. Cheung, "An Efficient Message Authentication Scheme for Link State Routing," Proc. 13th Ann. Computer Security Applications Conf. (ACSAC '97), pp. 90-98, 1997.

[10] M.T. Goodrich, Efficient and Secure Network Routing Algorithms, provisional patent filing, Jan. 2001.

[11] R. Perlman, "Network Layer Protocols with Byzantine Robustness," PhD dissertation, MIT LCS TR-429, Oct. 1988.

[12] V.N. Padmanabhan and D. Simon, "Secure Traceroute to Detect Faulty or Malicious Routing," SIGCOMM Computer Comm. Rev., vol. 33, no. 1, pp. 77-82, 2003.

◈◈◈