RIPPS: Rogue Identifying Packet Payload Slicer Detecting Unauthorized Wireless Hosts Through Network Traffic Conditioning

CHAD D. MANO, ANDREW BLAICH, QI LIAO, YINGXIN JIANG,
DAVID A. CIESLAK, DAVID C. SALYERS and AARON STRIEGEL
Department of Computer Science & Engineering, University of Notre Dame

Wireless network access has become an integral part of computing both at home and at the workplace. The convenience of wireless network access at work may be extremely beneficial to employees, but can be a burden to network security personnel. This burden is magnified by the threat of inexpensive wireless access points being installed in a network without the knowledge of network administrators. These devices, termed *Rogue Wireless Access Points*, may allow a malicious outsider to access valuable network resources, including confidential communication and other stored data. For this reason, wireless connectivity detection is an essential capability, but remains a difficult problem. We present a method of detecting wireless hosts using a local RTT metric and a novel packet payload slicing technique. The local RTT metric provides the means to identify physical transmission media while packet payload slicing conditions network traffic to enhance the accuracy of the detections. Most importantly, the packet payload slicing method is transparent to both clients and servers and does not require direct communication between the monitoring system and monitored hosts.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network Communications*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms: Management, Measurement, Security

Additional Key Words and Phrases: Network Security, Rogue Systems, Traffic Conditioning

1. INTRODUCTION

Computer security is a critical component of business operations for companies ranging from small businesses to international conglomerates. The threat of a malicious intruder forces network administration personnel to devote a significant portion of time to the detection of unauthorized users and the securing of network resources. Success is dependent on the vigilant deployment of security devices such

Author's address: C. D. Mano, Utah State University, Department of Computer Science, A. Blaich, Q. Liao, Y. Jiang, D. A. Cieslak, D. C. Salyers, A. D. Striegel, University of Notre Dame, Department of Computer Science and Engineering, Notre Dame, IN 46556.
This research was supported in part by the National Science Foundation through the grant CNS03-47392.

as firewalls, virus scanners, intrusion detection systems, and the ability of those devices to quickly and accurately identify malicious guests.

Wireless access points (WAPs) represent an important device which network administrators must be prepared to guard against. An employee with basic computer skills can purchase an inexpensive WAP and quickly configure/install it into a corporate network. Basic protection provided by MAC filtering can be easily subverted by MAC spoofing, a common feature of WAPs, enabling simple integration into the wired network. From a security perspective, this scenario is extremely dangerous as it creates a "backdoor" opening for malicious parties to gain access to the network. The danger is compounded by the fact that system administrators may be totally unaware that the vulnerability even exists. This type of unauthorized WAP is one type of *Rogue WAP* (RWAP) due to its ability to hide within an existing network and its potential for supporting mischievous activity.

Techniques for wireless detection can be broadly classified into *over the air* and *on the wire* [Henning 2003]. For example, currently proposed techniques range from wide-scale antenna deployment to wire-based host profiling to network communication analysis. Antenna-based systems are extremely expensive as they require full-coverage of the company campus. Many wire-based systems rely on MAC filtering which, as mentioned previously, can be subverted through spoofing. An alternative method is active probing which requires hosts to respond to communication attempts. Active probing techniques suffer as results may be inconclusive or inaccurate in cases where probed systems are configured to respond incorrectly, or to not respond at all to probing attempts. In particular, false alarms would be a problem as an authorized system which is down would appear identical to an unauthorized system which refuses to respond to probing requests.

Passive monitoring is a desirable method as it does not depend on host configuration settings which may be designed to easily avoid detection by active techniques. However, data resulting from purely passive observation techniques is often noisy, making highly accurate analysis difficult. Ideally, an RWAP detection method would utilize a monitoring technique which incorporates the "passive-like" monitoring characteristic of being independent of host settings and an "active-like" monitoring characteristic of producing highly accurate results.

This paper posits the question: *is it possible to deduce wireless connectivity and hence potential RWAP activity from using existing wired network traffic in a fast and accurate manner?* Hence, the premise of this paper is to identify of unauthorized and/or unsecured WAP (i.e. RWAP) connectivity without a wireless sensor or host-based deployment. To that end, we present the *Rogue Identifying Packet Payload Slicer* (RIPPS) system that combines an active network traffic conditioning technique with passive packet timing analysis to accomplish those goals. RIPPS incorporates a novel packet payload slicing technique to perform network traffic conditioning for significantly enhancing the accuracy and speed of its measurements. Moreover, the RIPPS conditioning technique manipulates existing traffic and does not require modifications to client systems nor the ability to communicate directly with these systems.

The remainder of the paper is organized as follows. Section 2 presents related work while Section 3 provides an overview of the RIPPS system and its components.

Section 4 describes experiments conducted with a fully functional prototype of the RIPPS system. Section 5 analyzes RIPPS with regards to deployment and overhead. Finally, Section 6 summarizes the study and offers several concluding remarks.

2. RELATED WORK

Wireless detection techniques can be classified as antenna-based [Chirumamilla and Ramamurthy 2003, Adya et al. 2004], or wire-based. Antenna-based techniques are available from a wide variety of vendors ranging from dedicated sensors to re-using existing host/AP wireless capabilities (wisEntry, Aruba Networks, AirMagnet, etc.). We limit our techniques to wire-based methods as antenna-based methods take an entirely different approach and have very different implementation requirements.

As shown in [Deraison and Gula 2003], system profiling applications including *nmap*, and *nessus* are able to identify a WAP through port scanning. *Rapids* is a commercial application based on similar techniques. These methods require active probing which can be defeated by a system that does not to respond to the probing attempts. This can be the result of a host system setting or even firewall or NAT devices which are often set, by default, to deny all incoming communication requests. Conversely, Bro [Handley et al. 2001] and other anomaly detection mechanisms could potentially rely on traffic observations (normalization of traffic, etc.) for WAP and/or NAT discovery. However, as will be shown in the later experiments, anomaly detection via traffic normalization/analysis can be inconclusive when faced with only short-lived connections and the advent of faster wireless connectivity.

MAC filtering applications including *APTools* identify systems by matching MAC addresses with vendor MAC address lists. The work in [Chirumamilla and Ramamurthy 2003] provides an infrastructure which relies on MAC address filtering to identify and prevent RWAPs. MAC spoofing is a simple technique, and a feature in most routers, which can be used to avoid detection by this method. Although techniques for detecting MAC spoofing as proposed in [Guo and Chiueh 2006] for wireless and others for the wired side (ex. 802.1X certificates) would negate the ability to place a new network element, such approaches have not yet seen widespread adoption.

Another related area of study is the detection of NAT-enabled devices, a common configuration of a WAP. *SFlow* detects NAT-based hosts by identifying anomalous TTL values from the router functionality of the NAT. In [Bellovin 2002], the authors use the ID field of the IP header to determine the number of hosts behind a NAT device. [Beverly 2004] uses a similar model but incorporates TTL, window size, and SYN size header information. Although these approaches are effective when multiple hosts are present behind a NAT device, they may fail when only a single device is used. Hence, a single host behind an RWAP (as would typically exist from an employee using it for personal use) would be nearly undetectable from these approaches. *SFlow* utilizes port number observation as NAT devices tend to bias towards the high side as to avoid server conflict.

In [Cheng and Marsic 2001], a similar RTT metric and fuzzy reasoning are utilized to detect wireless connections in a client/server environment for the purpose of improving QoS. This technique relies on the differences in the average RTT measured from the client perspective as well as the standard deviation of RTTs. Equipment used in the study supported a throughput of 1.6Mb/s, much lower than the standards of today. Our experiments show that the RTT and associated standard deviations using currently available equipment does not provide a distinct separation between wired and wireless hosts.

In [Wei et al. 2005], the authors present an active monitoring scheme for the classification of access network types. The active technique works by requesting a remote host to send numerous packet pairs and measuring the spacing of the packets upon arrival. A large time gap is indicative of a slow medium such as dial-up Internet access while a small gap indicates a faster medium such as Ethernet. This method provides accurate results but depends on the ability to communicate with the remote host and trusting the host to respond appropriately to packet pair transmission requests.

[Beyah et al. 2004] presented a passive method of detecting RWAPs in a network. This method is based on the observation of inter-packet spacing differences between wired and wireless networks. While their results are promising, it is not entirely clear as to the network speeds of the hardware used. The results lead us to believe that the wired hosts were connected via a 100Mb/s link while the wireless hosts utilized an 802.11b type of connection. As our experiments will show, faster wireless (802.11g, 802.11 draft-n) and OS diversity make it difficulty to cleanly distinguish between wired and wireless hosts. In a similar vein, the recent work in [Wei et al. 2006] examines TCP ACK pairing from a theoretical and limited experimental perspective for both 802.11b and 802.11g. However, OS diversity (specifically Windows) also poses issues as noted later in our experimental studies. These two methods are the only other transparent traffic-based methods designed specifically for wireless connectivity detection of which we are aware.

3. SYSTEM OVERVIEW

The RIPPS system acts as a pass-through device for all network traffic while selectively monitoring network hosts. It effectively identifies the presence of a local wireless link between two hosts that have a communication path that passes through the RIPPS device. A RIPPS system could be incorporated at any point in a network to protect important network resources. For example, a RIPPS system placed in front of a corporate data server would serve to protect vital company information from an intruder utilizing an unauthorized and/or unsecured wireless connection from an RWAP inside the company perimeter. Figure 1 illustrates a typical deployment scenario for RIPPS.

The RIPPS system is comprised of several distinct components that work together to quickly and accurately detect the presence of local wireless transmission media. The actual wireless detection is accomplished through an analysis of LRTT measurements derived from TCP communication which passes through the RIPPS system. Importantly, this process is enhanced by the second component (slicing) that conditions network traffic to help by both removing interfering noise and expediting medium classification. In short, RIPPS offers the desirable transparency

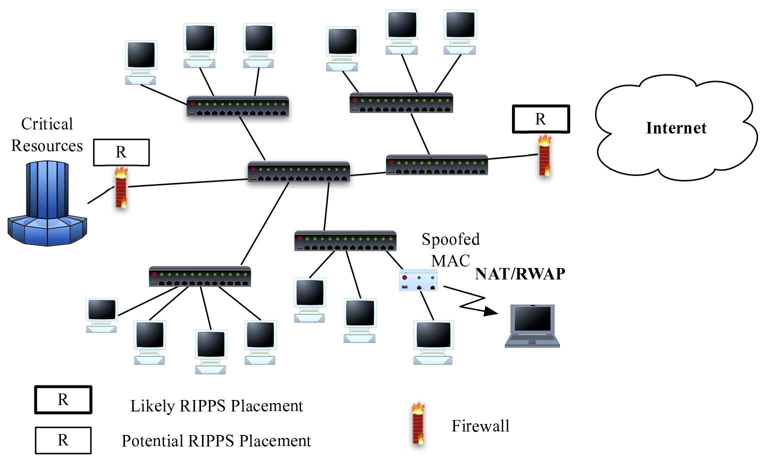


Fig. 1. Example RIPPS environment

and deployment characteristics of purely passive approaches (no changes or installed agents at clients), the accuracy of active probing approaches, and expedited measurement mechanisms.

It is important to note that the active components of the RIPPS process are on-demand mechanisms that can be triggered through either periodic monitoring or by other external sensing mechanisms. Moreover, the performance impact of RIPPS can be strictly limited as discussed in Section 5.2. RIPPS can act in a stand-alone manner or be incorporated into existing approaches such as antenna-based solutions for additional accuracy and/or localized host isolation (i.e. SNMP exchange with switches and/or routers). The following subsections analyze the key RIPPS components in greater detail with regards to the Local Round Trip Time (LRTT), packet slicing, and packet ordering.

3.1 Local Round Trip Time

In a broad sense, the latency of network based communication can be viewed as the result of either WAN-side or LAN-side effects. WAN-side latency is the result of many factors which can vary significantly between communication sessions and especially between differing communication host pairings. On the other hand, LAN-side latency, represents a reasonably consistent path between the end host and WAN gateway. Therefore, we limit ourselves to the RTT associated only with LAN-side traffic in order to remove WAN-side variations and emphasize the connectivity medium of hosts in the LAN.

Local round-trip time is a measurement of the time delay between a message to and response from a specific host in the LAN. A sensor placed at the edge of the LAN (or other appropriate location) collects data in a passive manner. The metric is obtained by categorizing packets based on a connection-wise basis (source IP, source port, destination IP, destination port) and storing both a timestamp as well as an expected acknowledgment (ACK) number. The timestamp associated with the messages is calculated solely by the sensor, hence the relative time between messages is consistent and free from time synchronization problems. Outbound packets are similarly classified on a connection-wise basis and the ACK number from the TCP header is compared to the expected ACK numbers calculated previously. The LRTT is the time difference between incoming packets and corresponding ACK packets as observed by the monitor itself.

The LRTT is influenced by a variety of factors. First and foremost, the metric is influenced by the transmission medium between the monitoring system and host. The purpose of RIPPS is to isolate this influence in order to accurately identify the transmission type.

Second, the LRTT metric is influenced by the size of the data payload. The variance in packet size results in varying LRTT values for a single host which may cause misleading results when comparing hosts to one another. This problem can be eliminated by calculating multiple LRTT values each based on packets of uniform size. LRTT values calculated with small packets are desirable in that small packets minimize the influence of bandwidth capabilities on packet timing metrics, while large packets maximize those effects [Cheng and Marsic 2001].

While our purpose is not to identify bandwidth capabilities, intuitively this may be useful because of the differences between wired and wireless capacities. However, the advent of MIMO (Multiple In/Multiple Out) and faster theoretical wireless connectivity are a cause of significant concern for traffic analysis techniques. Furthermore, large packets are typically a significant portion of the overall traffic contained in a typical network. Ignoring this portion means wasting data, thereby potentially limiting the speed in which transmission media can be identified. In addition, a typical client system receives relatively large data transmissions while sending only small requests. Therefore, ignoring large packets may result in overlooking the activity of potentially dangerous client systems.

However, while large packets maximize the influence of bandwidth on LRTT (a desirable outcome for slower wireless speeds), large packets also introduce significant noise[1]. RIPPS remedies the noise issue of large packets by slicing the larger packets into smaller, ideally-sized packets through TCP-level modifications as will be discussed later. This core contribution of our paper enables not only the inclusion of more packets, but also greatly increases the speed and quality at which a determination of transmission type can be made.

In addition, there exist other contributing factors to the overall LRTT value which must be addressed. First, LAN-side jitter effects (congestion, queuing, etc.) are present in the measurement, but are assumed to be a minimal/reasonable component, resulting in precise measurements which can be used to identify relative LRTT differences of LAN devices[2].

A second factor is the *retransmission ambiguity* problem, presented in [Karn and Partridge 1991]. Simply put, the problem is based on the fact that if a packet is retransmitted then it is ambiguous which transmission the corresponding acknowl-

[1]For the purposes of streamlining this paper, the reader is referred to [Mano 2006], with a discussion of the shortcomings of using purely passive LRTT measurements at an enterprise (campus gateway) level with thousands of hosts.
[2]The experimental studies note the tolerance of RIPPS to various forms of congestion in multi-tiered networks.

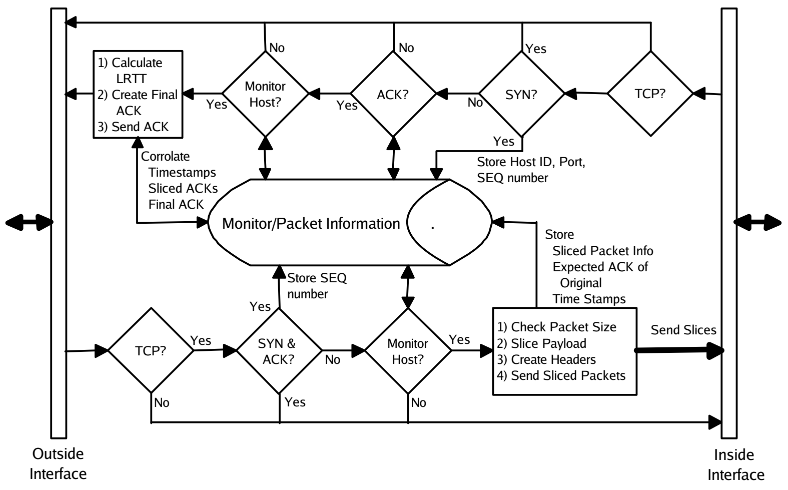


Fig. 2. Processing flow of packets for the packet slicing system.

edgment packet refers to. We negate these potential affects by using Karn's method of discarding LRTT values which may suffer from the ambiguity problem.

A final factor is the absolute time required for the host to process incoming messages before sending an acknowledgment. While the original RFC for TCP include strong wording regarding what the ACK represents, the guidelines for ACK responsiveness and generation are left vague. While the vagueness can create significant issues for utilizing a purely passive LRTT, the next subsections offer techniques for minimizing the above problem through slicing and intelligent packet re-ordering.

3.2 Packet Slicing

Packet slicing is a network traffic conditioning technique which serves as the key component of RIPPS. This technique has three vital properties which make it an interesting solution for wireless connectivity assessment. First, packet slicing dramatically increases the percentage of network data which can be accurately utilized in assessing the connectivity type. Second, it maintains consistency with in-band monitoring techniques as direct communication with suspect hosts is not required. Finally, packet slicing eliminates the temporal spacing problem (i.e. small TCP data packets are located infrequently together), resulting in an increased ability to quickly identify wireless devices.

The general concept of packet slicing is simple, yet it improves transmission medium identification capabilities many fold. As noted in various network traffic studies, very small (64 byte) and very large packets (approaching MTU) make up the vast majority of network traffic. However, from the perspective of an individual client, the percentage of small incoming packets is much less than the percentage of all traffic. The prevalence of large downstream (incoming) and small upstream (outgoing) packets can be directly attributed to typical TCP functionality. The goal of RIPPS is to take these large ingress data packets and condition them for use in LRTT metrics by slicing payloads and creating many new smaller packets.

To describe the implementation of packet slicing, the processing of a single flow from SYN to FIN is illustrated in Figure 2. A client, A, requests a connection with a server, B, by sending a SYN packet. RIPPS identifies the SYN packet and notes the source host, sequence (SEQ) number, and the port number. B responds with a SYN/ACK, followed by an ACK from A, completing the handshake. Once communication is established, RIPPS monitors incoming traffic for packets with the appropriate destination address and port numbers. If the host is to be actively monitored, packets of a sufficient size are sliced into multiple packets and appropriately shaped. If the host is not being monitored or the packet size is too small, the sequence number is noted and the packet sent onwards with consideration for order preservation.

In short, packet payload slicing spreads a single payload over multiple packets, attaching each payload slice to an appropriate header. The headers from the original packet are used to easily create valid headers for each new packet. The Ethernet header is unchanged from the original, and the IP and TCP headers are modified slightly to validate the newly created packet. After modifying appropriate header fields, including checksums, the slices are re-ordered, shaped, and forwarded. From the viewpoint of any applications running on host A, data is still delivered as normal from B. From the perspective of B, the normal level of ACKs are received from A as RIPPS will take special care to hide its presence from B.

As noted earlier, it is important to emphasize that *the packet slicing activity of RIPPS is not always on*, but is used periodically by intelligently selecting which hosts should be monitored and when monitoring should be performed (see 5.2). When coupled with a hybrid approach (minimal impact first pass followed by aggressive assessment if necessary), the impact on individual host systems which are being monitored and on the overall network is minimal.

3.2.1 *Slice Ordering.* While slicing increases the number of potential measurement points, the host is not obligated to acknowledge each packet. Critically, slices that do not result in additional measurement points represent potential wasted overhead.

Our internal experiments show different operating systems providing ratios varying from one to one and two to one depending on inter-packet spacing (Windows, Linux) with others cumulatively sending ACKs in a time-wise manner (Mac OS). Hence, operating systems that bias heavily towards cumulative ACKs would invalidate the entire foundation of RIPPS. To that end, we incorporate the packet reordering techniques from TCP Sting [Savage 1999]. In short, TCP Sting re-orders a group of TCP data packets $(P_0, P_1, \ldots, P_{N-1})$ to $(P_1, \ldots P_{N-1}, P_0)$ in order to force the creation of ACK messages for monitoring courtesy of the TCP Fast Retransmit mechanism.

While this mechanism is highly successful when coupled with RIPPS to force ACK generation, it does introduce the potential for additional ACK ambiguity. For example, consider the earlier case of a set of sliced packets P_0 through P_{N-1} re-ordered in the prescribed Sting ordering (P_0 is transmitted last). As a result, the client will generate Fast Re-Transmit ACKs containing an ACK with the sequence number of P_0 (i.e. cumulatively acknowledging everything before P_0). Hence, potentially $N-1$ packets will contain that same ACK. Aside from the cases of

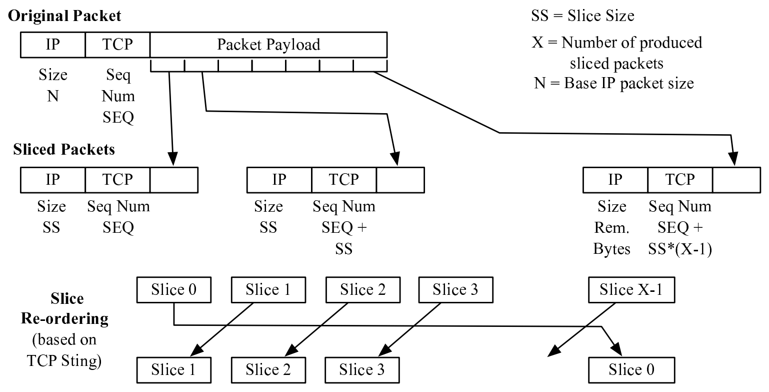


Fig. 3. RIPPS Slicing and Re-ordering

retransmissions, such ambiguity is not a problem as the TCP Fast Re-Transmit behaviors are relatively consistent across the default OS network stacks. Moreover, extended functionality such as SACK is not a problem provided that the ACK is still generated according to TCP Fast Re-Transmit behavior.

Since RIPPS is operating transparent to both the client and server (application-wise), it is essential that RIPPS appropriately intercept and squelch returning ACKs (forced by re-ordering). With the slicing of each data packet, RIPPS notes the expected acknowledgment and a squelch (gobble) / no squelch flag to denote if the packet should be released from RIPPS. Packets that are to be released to the server (i.e. the cumulative ACK that is generated based on Slice 0) are released appropriately. As will be noted in the experiments, the response time of the cumulative ACK varies considerably between different operating systems while the TCP Fast Re-transmit forced by Sting is relatively fast and accurate. The effects of employing TCP Sting versus RIPPS without re-ordering and the effect of cumulative ACKs are examined in the next section of experimental studies.

4. EXPERIMENTAL STUDIES

For our experimental studies, a fully functional prototype of RIPPS was implemented. The prototype was developed in RedHat Enterprise 4 utilizing the *libpcap* libraries for simplicity of development. The RIPPS monitoring system was a dual-core Opteron workstation with three network ports, *eth0* for SSH/external management and a PCI-X Intel dual Gigabit server adapter providing the pass-through monitoring ports of *eth1* and *eth2*.

The network testbed utilized in the experiments is shown in Figure 4. Multiple hosts were connected through the pass-through RIPPS box to the university LAN. LRTT values were computed relative to the RIPPS box. Optionally, traffic congestion was introduced by hosts at the H_1, H_2, and H_3 positions to provide uni-directional and bi-directional congestion both visible and invisible to RIPPS.

The hosts for comparison were connected at the testing points noted in the figure to vary between wired, wired-NAT, and wireless connectivity. A wide variety of host/network configurations were tested incorporating Windows, Linux, and Mac

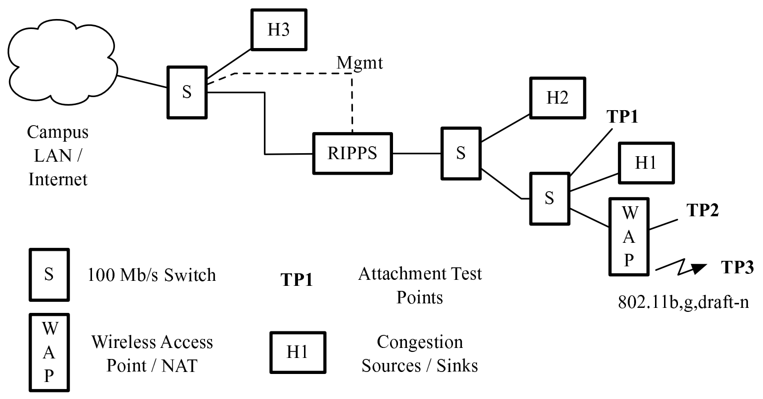


Fig. 4. RIPPS Testbed.

OS as operating systems and 100 Mb/s Fast Ethernet (TP_1), 100 Mb/s Fast Ethernet behind a NAT (TP_2), and wireless 802.11b, g, and draft-n (TP_3). Connectivity was provided through a D-Link DIR 625 draft-n wireless router. Multiple applications (web, ftp, ssh, scp) were compared for completeness. Wireless connectivity to the WAP was strictly limited between b, g, and draft-n in order to not force compatibility modes. Considerations for 10 Mb/s Ethernet and other wireless routers are available in [Mano 2006].

In the experiments, two key performance metrics were assessed:

—*Medium separation:* Regardless of the application involved, was there a clear separation between the wired and wireless monitored characteristics? Separation was measured by the 95% confidence interval (CI) of the measured average LRTT.

—*Time to classify:* If a clear separation could be identified, at what time could a reasonable classification be trusted? Classification speed was validated through short-lived web (CNN, Yahoo) and ssh (login, ls, logout) connections.

4.1 Baseline - No RIPPS

To start, we analyzed the performance that occurs when RIPPS is not used. A summary table is provided in Table I of the various OS/network combinations. Packets were passed through the RIPPS box but the packets themselves were only noted for LRTT. Neither slicing, re-ordering, nor shaping was applied to the packets. Results were computed over several runs with cache flushes[3].

From the table, several key observations can be noted. First, in the case of Windows and web traffic (likely the most common occurrence), discrimination between wired and wireless connectivity cannot be done with a degree of confidence. While there is limited separation for the wired versus wireless cases, the results are inconclusive statistically. Moreover, as these results were conducted in an idle setting maximizing differentiation, congestion could introduce small amounts of bias to the readings, removing whatever limited separation exists in the baseline case. We

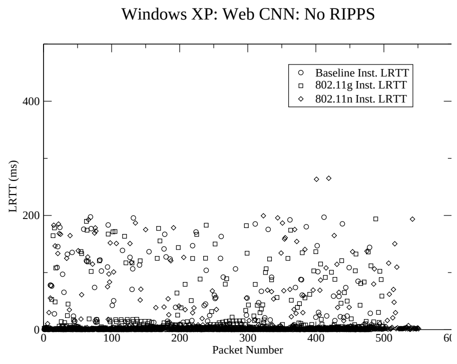
[3]Variations in packet counts and CPU impact occurred due to changing content at the websites, ex. different ads or articles.

OS	**Application**	**Network Type**	**Matched ACKs**	**Avg. LRTT (ms) +/- 95% CI**	**Std. Dev. of LRTT**
Windows	web_cnn	wired	481	17.508 +/- 3.213	42.836
Windows	web_cnn	wireless_g	499	20.802 +/- 3.026	41.087
Windows	web_cnn	wireless_n	544	18.606 +/- 3.022	42.836
Linux	web_cnn	wired	436	2.000 +/- 0.496	6.299
Linux	web_cnn	wireless_b	344	11.520 +/- 0.882	9.939
Mac	web_cnn	wired	325	4.038 +/- 1.826	20.011
Mac	web_cnn	wireless_g	358	11.010 +/- 3.435	39.504
Windows	web_yahoo	wired	111	14.022 +/- 5.820	37.272
Windows	web_yahoo	wireless_g	127	18.923 +/- 6.099	41.776
Windows	web_yahoo	wireless_n	107	15.231 +/- 5.506	34.617
Windows	web_nd	wired	37	15.058 +/- 12.893	47.668
Windows	web_nd	wireless_g	39	32.560 +/- 14.037	53.281
Windows	web_nd	wireless_n	32	24.228 +/- 14.086	48.433
Windows	ssh	wired	33	62.141 +/- 18.518	64.659
Windows	ssh	wireless_g	35	67.438 +/- 19.085	68.630
Windows	ssh	wireless_n	33	71.138 +/- 19.461	67.952
Linux	ssh	wired	53	2.144 +/- 1.752	7.754
Linux	ssh	wireless_b	54	5.095 +/- 1.758	7.851
Mac	ssh	wired	41	0.251 +/- 0.018	0.069
Mac	ssh	wireless_g	35	7.286 +/- 9.046	32.528
Windows	scp	wired	2782	0.723 +/- 0.133	4.266
Windows	scp	wireless_g	3060	16.905 +/- 0.268	9.013
Windows	scp	wireless_n	2697	4.185 +/- 0.204	6.437
Linux	scp	wired	2192	0.641 +/- 0.060	1.712
Linux	scp	wireless_b	2776	106.762 +/- 1.133	36.289
Mac	scp	wired	1215	0.573 +/- 0.010	0.221
Mac	scp	wireless_g	2711	77.620 +/- 1.362	43.096
Windows	ftp	wired	5621	0.598 +/- 0.068	3.080
Windows	ftp	wireless_g	9147	19.088 +/- 0.311	18.100
Windows	ftp	wireless_n	6825	2.890 +/- 0.109	5.477
Linux	ftp	wired	4313	1.016 +/- 0.105	4.177
Linux	ftp	wireless_b	4717	69.544 +/- 0.936	39.061
Mac	ftp	wired	3828	0.719 +/- 0.096	3.608
Mac	ftp	wireless_g	4413	4.658 +/- 0.322	13.001

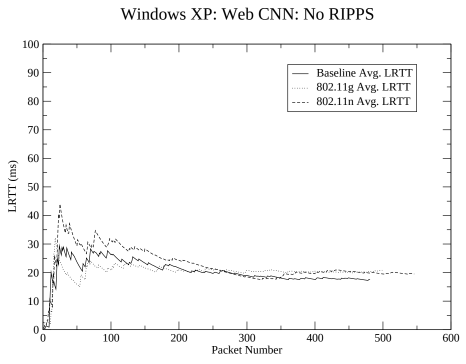
Table I. Baseline case - no RIPPS

also note that the lack of clear separation manifests itself across multiple websites (Akamized or not, small versus many sites, etc.). The lack of separation manifests itself most clearly in short-lived connections (web, ssh). Although reasonable separation occurs with Linux and Mac OS X-based hosts, these hosts are typically in the minority rather than majority in the enterprise. Experiments were not conducted with draft-n on Linux or Mac OS X as capable drivers did not exist.

A second observation is that as the number of packets measured increases such as with a secure file copy or FTP, the accuracy of the measurement does improve significantly. However, the number of packets alone is insufficient as the lack of separation manifests itself for the short-lived connections in each session (see Figure 5 and Figure 6). In short, the noise will reappear at similar levels in each new session,

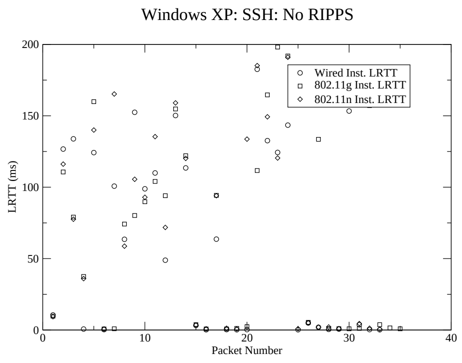


(a) Instantaneous LRTT.

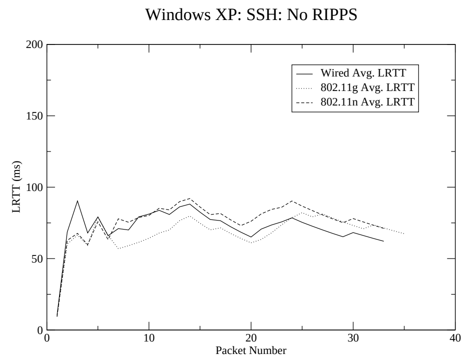


(b) Cumulative average LRTT.

Fig. 5. Per-packet analysis of LRTT without RIPPS on web traffic.



(a) Instantaneous LRTT.



(b) Cumulative average LRTT.

Fig. 6. Per-packet analysis of LRTT without RIPPS on simple ssh traffic.

regardless of monitor length. The graphs in each figure show both the instantaneous LRTT measurement (shown via scatterplot) and cumulative average LRTT for the most problematic cases of web access and SSH. We define the cumulative average LRTT to be the computed LRTT at that specific measurement point (i.e. the 3rd packet represents the average of packets 1-3).

As the network itself is idle, one can only conclude that the variation in delay originates from the OS (Windows) network stack itself must not be forcing out immediate ACKs. Similar LRTT observations under Windows were noted using alternate browsers such as Firefox. In addition, larger spikes tended to be correlated with the PSH flag being set. Moreover, the performance in the tests lends to the conclusion that the medium itself is not being stressed with the short-lived connections. The short-lived characteristic of the connections (ssh, web) tends to finish the connection before the TCP slow start window expands to generate sufficient traffic. While the cumulative average LRTT does eventually settle, a degree of statistical confidence in the separation of the measurements cannot be

OS	**Application**	**Network Type**	**Matched ACKs**	**Avg. LRTT (ms) +/- 95% CI**	**Std. Dev. of LRTT**
Windows	web_cnn	wired	6705	0.471 +/- 0.004	0.195
Windows	web_cnn	wireless_g	7020	48.802 +/- 0.907	46.192
Windows	web_cnn	wireless_n	6295	4.800 +/- 0.209	10.056
Linux	web_cnn	wired	5205	0.384 +/- 0.003	0.142
Linux	web_cnn	wireless_b	5470	130.102 +/- 1.277	57.398
Mac	web_cnn	wired	7543	0.458 +/- 0.014	0.739
Mac	web_cnn	wireless_g	6052	47.708 +/- 0.854	40.391
Windows	web_yahoo	wired	466	0.390 +/- 0.009	0.119
Windows	web_yahoo	wireless_g	462	58.712 +/- 1.689	22.070
Windows	web_yahoo	wireless_n	482	6.617 +/- 0.203	2.707
Windows	web_nd	wired	834	0.346 +/- 0.005	0.091
Windows	web_nd	wireless_g	927	106.876 +/- 3.082	57.044
Windows	web_nd	wireless_n	587	7.020 +/- 0.372	5.476
Windows	ssh	wired	39	0.260 +/- 0.023	0.086
Windows	ssh	wireless_g	37	3.249 +/- 0.466	1.723
Windows	ssh	wireless_n	36	2.564 +/- 0.535	1.950
Linux	ssh	wired	56	0.301 +/- 0.030	0.136
Linux	ssh	wireless_b	57	12.805 +/- 1.660	7.615
Mac	ssh	wired	49	0.314 +/- 0.015	0.063
Mac	ssh	wireless_g	49	6.445 +/- 1.447	6.158
Windows	scp	wired	111628	0.955 +/- 0.004	0.848
Windows	scp	wireless_g	112296	149.913 +/- 0.415	84.606
Windows	scp	wireless_n	88608	8.167 +/- 0.055	9.960
Linux	scp	wired	128860	0.798 +/- 0.002	0.384
Linux	scp	wireless_b	129135	141.172 +/- 0.074	16.097
Mac	scp	wired	116992	0.580 +/- 0.004	0.872
Mac	scp	wireless_g	50819	73.291 +/- 0.706	96.704
Windows	ftp	wired	199754	0.849 +/- 0.003	0.682
Windows	ftp	wireless_g	213109	283.607 +/- 0.495	138.761
Windows	ftp	wireless_n	187634	5.693 +/- 0.049	12.813
Linux	ftp	wired	226023	0.948 +/- 0.002	0.600
Linux	ftp	wireless_b	228814	65.093 +/- 0.048	13.867
Mac	ftp	wired	216274	0.487 +/- 0.001	0.209
Mac	ftp	wireless_g	190355	92.789 +/- 0.413	109.576

Table II. The performance of RIPPS

reached. Furthermore, a premature assessment (time-based or packet-based) could result in incorrect classification of the underlying media.

The results of this baseline experiment clearly illustrate the need for RIPPS if one desires to use LRTT as a medium classification mechanism. The simple LRTT mechanism fails in the web and ssh cases for both medium separation and time to classify. While the works in [Beyah et al. 2004] and [Wei et al. 2006] showed promise, we suspect the likely usage of UNIX-based hosts, long-lived TCP connections, and slower, half-duplex wireless mediums (802.11b and 802.11g) biased the measurements in their work significantly.

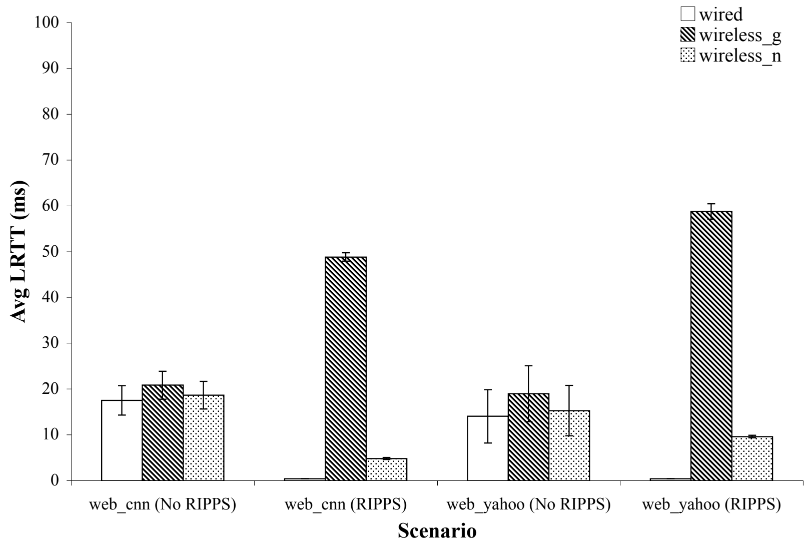


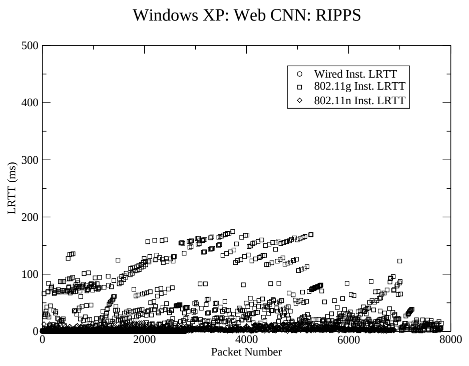
Fig. 7. Comparison of RIPPS versus no RIPPS for Windows, web

4.2 Comparing RIPPS

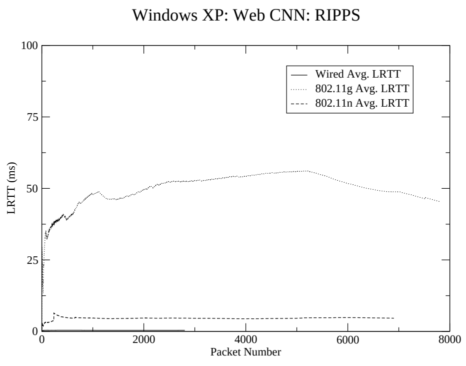
Table II shows the performance when RIPPS is engaged in the same scenarios. RIPPS was enabled with a slice size of 60 bytes (approximately 100 byte L3-L7 packet size), a minimal 1 microsecond shaping between sliced packets, and a 1 millisecond spacing between groups of sliced packets. Figure 7 shows the performance (no RIPPS) versus RIPPS in a side by side graph.

Several key observations stand out from the table. First, the separation of the various medium is now visible in all cases, regardless of application. This separation can be attributed to two factors, namely the uniformity of the measurement packet size and the number of measurement points. While the perceived rendering/receipt delay from the user standpoint was quite similar (web traffic), the number of observed points increased dramatically in the same approximate amount of time. We also note that despite the presence of MIMO for draft-n, the separation is still quite noticeable versus the wired baseline case.

Figure 8 breaks down this trend further by showing the instantaneous and accumulated average values of a standard web access under RIPPS. The breakdown of instantaneous values further reaffirms the uniformity of measurement Despite having the same relatively small number of incoming data packets, RIPPS identifies significantly more LRTT measurements. Although the additional measurement points introduce overhead, the above table represents an extreme case whereby



(a) Instantaneous LRTT (downsampled).



(b) Cumulative average LRTT.

Fig. 8. Per-packet analysis of LRTT with RIPPS on web traffic..

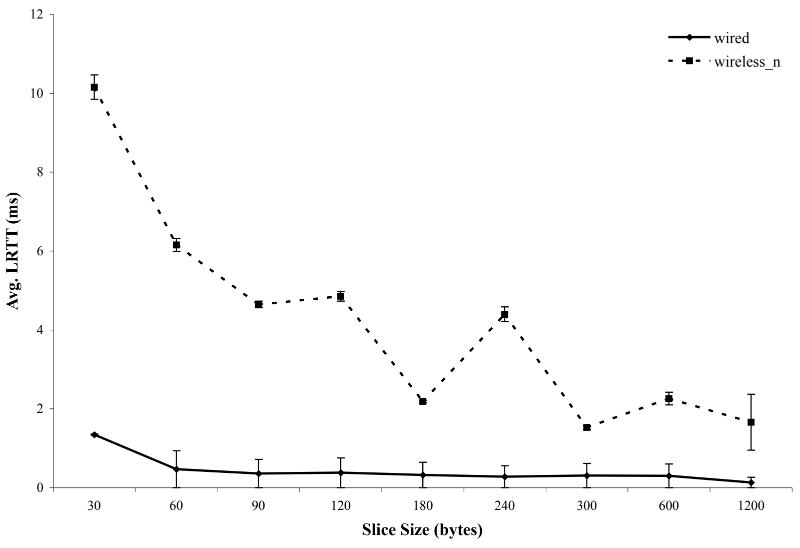


Fig. 9. Effect of varying slice sizes on RIPPS (Windows, web_cnn)

Slice Delay	**Network Type**	**Matched ACKs**	**Avg. LRTT (ms) +/- 95% CI**	**Std. Dev. of LRTT**
0	wired	6328	0.996 +/- 0.015	0.719
1	wired	2955	0.469 +/- 0.006	0.203
100	wired	7053	0.196 +/- 0.003	0.128
1000	wired	2245	0.300 +/- 0.009	0.267
0	wireless_n	2528	9.650 +/- 0.387	11.825
1	wireless_n	5770	6.151 +/- 0.169	7.783
100	wireless_n	6381	6.702 +/- 0.120	5.838
1000	wireless_n	4971	2.641 +/- 0.059	2.531

Table III. Effect of inter-slice delay (shaping) on RIPPS (Windows, web_cnn)

Sting Enabled?	**Network type**	**Matched ACKs**	**Avg. LRTT (ms) +/- 95% CI**	**Std. Dev. of LRTT**
Yes	wired	2744	0.393 +/- 0.004	0.120
Yes	wireless_n	6295	4.800 +/- 0.209	10.056
No	wired	1912	6.756 +/- 5.890	156.549
No	wireless_n	1153	14.083 +/- 9.811	202.499

Table IV. Effect of packet re-ordering on RIPPS (Windows, web_cnn)

RIPPS is permanently enabled. In a deployment scenario, RIPPS would likely be set up to employ a hybrid approach (slice first 1 or 2 packets if host is up for monitoring and then conduct aggressive RIPPS as necessary).

4.3 Varying RIPPS

Figure 9 shows the effect on the observed average LRTT as the slice size (TCP payload) is varied from 30 bytes to 1200 bytes on solely the Windows wired and wireless draft-n scenarios. As noted in the figure, an increase in the packet slice size decreases the relative accuracy of RIPPS. Put simply, the LRTT becomes noisier due to the larger discrepancy in downstream versus upstream packet size (data versus ACK) and due to the net reduction in the number of monitoring packets.

Table III notes the effect on the observed average LRTT as the slice shaping delay is varied from no delay to one millisecond (1000 microseconds). With a significant enough shaping delay, the medium access itself is taxed less, slackening the separation although not significantly enough. While the 100 microsecond delay offers the 'cleanest' signal in terms of the wired medium, a low slice size with such a delay creates a noticeable impact on web page load times. In contrast, the 1 microsecond delay offers similar performance with no perceptible delay in page load times.

Finally, Table IV notes the effect of packet re-ordering on RIPPS. Table IV notes the necessity of forcing the packet retransmit and avoiding the network stack bottlenecks for an accurate measurement.

4.4 Congestion & Mixed Signal Sources

Although the previous subsection examined a pure wired or wireless case, it is extremely likely that a RWAP would have both wired and wireless hosts connected,

If RIPPS	Application	Network type	Matched ACKs	Avg. LRTT (ms) +/- 95% CI	Std. Dev. of LRTT
no	web_cnn	wired_NAT	608	24.139 +/- 3.070	46.005
no	web_cnn	wireless_g	799	20.432 +/- 2.454	42.162
no	web_cnn	wireless_n	615	23.842 +/- 2.986	45.011
yes	web_cnn	wired_NAT	12541	1.547 +/- 0.019	1.286
yes	web_cnn	wireless_g	16662	28.169 +/- 0.504	39.548
yes	web_cnn	wireless_n	11593	4.085 +/- 0.058	3.779

Table V. The performance of no RIPPS and RIPPS with mixed signal sources (Windows)

Network Type	Congestion Type	Matched ACKs	Avg. LRTT (ms) +/- 95% CI	Std. Dev. of LRTT
wired	down $H3 - H1$	6944	0.54 +/- 0.004	0.187
wired	up+down $H3 = H1$	6529	0.576 +/- 0.005	0.197
wired	down $H2 - H1$	5608	0.577 +/- 0.0151	0.577
wired	up+down $H2 = H1$	4745	2.133 +/- 0.0543	1.907
wireless_n	down $H3 - H1$	6293	5.994 +/- 0.1654	6.694
wireless_n	up+down $H3 = H1$	5901	6.841 +/- 0.2578	10.106
wireless_n	down $H2 - H1$	6134	5.398 +/- 0.09812	3.921
wireless_n	up+down $H2 = H1$	4686	11.133 +/- 0.5257	18.362

Table VI. Performance of RIPPS under VBR congestion (Windows, web_cnn)

hence exhibiting a mixed signal characteristic. Table V shows the results of only the web access in the mixed environment where one wired Windows host is mixed with the other host listed in the table (two devices, one wired, one variable access methods). In the table, we assume that both hosts each conduct the same access pattern (retrieve the contents of a web page, i.e. CNN). Even in the mixed signal case, RIPPS is still able to clearly differentiate between the MIMO draft-n and the wired host.

Finally, Table VI notes the effect of congestion on the monitored RIPPS behavior. In the congestion case, UDP traffic is streamed in VBR bursts to achieve an average rate of 70 Mb/s between various hosts noted in the earlier experimental testbed. Congestion is induced to both appear invisible ($H1$ to $H2$) and visible ($H1$ to $H3$) to the RIPPS monitoring system. The goal of the experiment was to validate the tolerance of RIPPS to multiple queuing levels such as may appear in a larger enterprise environment. We note that with additional congestion (upstream and downstream versus pure downstream), the gap is not only consistent but actually widens for similar applications. While this test is not necesarily indicative of all environments, we believe it lends credence to reasonable congestion tolerance for RIPPS. In the event of extremely complex environments or non-wired links, the monitoring point for RIPPS would likely be pushed closer to the host to reduce the effect of tremendously varying links hidden from RIPPS.

5. ANALYSIS

5.1 Network Deployment

An enterprise RIPPS must provide an automated calibration system, as well as an intelligent method of scanning network hosts. A calibration system is necessary as the strength of LRTT metrics lie in the relation between values generated through differing transmission media. Calibration occurs by the RIPPS system systematically monitoring authenticated known hosts. In the simplest case, only a single host (i.e. the management station) could be used as the calibration host. LRTT values from these known hosts serve as a baseline for the remaining hosts.

For deployment considerations, we assume a network environment whereby only MAC-level filtering is employed (valid host yields a valid IP via DHCP lease). As the past behavior of a host may not be indicative of its current state, periodic network scanning is vital. Consider the case where a wired host is identified by RIPPS and shortly thereafter is disconnected from the network. Then, a RWAP spoofing the MAC address of the removed host is inserted in place of the wired host. It is highly probable that the spoofed MAC address causes the RWAP to receive the same IP address as the previous host. If this IP address is considered safe by RIPPS and is therefore not scanned again until the session time expires, the RWAP will avoid detection.

Hence, an effective method of identifying hosts to be monitored must encompass two goals. First, each host is to be monitored periodically, but at random intervals. The second goal is to monitor all hosts, including those with a very short window of activity. The malicious intruder will likely minimize exposure time on the network, quickly accomplishing his or her goal and exiting. Missing such an event would nullify the protection the RIPPS system offers.

When a SYN packet is observed, RIPPS immediately logs the appropriate information to begin monitoring the host as described previously. RIPPS probing is conducted on a two stage basis. In the first stage, *quick assessment*, hosts are examined with minor applications of RIPPS. For example, each host might be monitored via the first stage every five minutes with one or two full size data packet(s) conditioned by RIPPS. These initial measurements formulate the basis of transitioning to stage two, *aggressive assessment*. Hosts in stage two have the entirety of their TCP packets monitored with RIPPS conditioning to a reasonable statistical threshold. The results from the aggressive assessment can then be tied into external mechanisms such as network isolation (VLAN), additional intrusion monitoring, and/or alert generation coupled with location information via existing network tools (SNMP for physical port identification). The number of hosts being concurrently chosen for the rough assessment can be limited either with a fixed impact with random selection or biased by the last monitoring time of a given host.

Table VII shows the measurements obtained by the quick assessment approach with a 95% confidence interval and the first captured flow of the Windows web_cnn test. In fact, the single slice case demonstrates reasonable separation even up to a 99.999% confidence interval between the wired and wireless cases for the first flow

Network Type	Avg LRTT (ms) Slice One Pkt	Avg LRTT (ms) Slice Two Pkts
wired	0.21 +/- 0.02	0.28 +/- 0.03
wireless_g	4.68 +/- 1.16	5.64 +/- 0.81
wireless_n	1.97 +/- 0.25	2.25 +/- 0.21

Table VII. First (quick) stage assessment of slicing only one or two full-size packets (Windows, web_cnn)

in the session. We note that the source of the quick assessment in both cases is only one or two incoming full-sized data packets.

5.2 RIPPS Overhead/Performance

The maintenance of a minimum level of overhead is critical to the functionality of the RIPPS system as additional network load can hamper communications, diminishing end user quality of service (QoS). Although one could leave RIPPS on permanently, such an approach is simply not necessary. Rather, the two tier process outlined earlier can limit the impact of RIPPS.

The overhead introduced by aggressive RIPPS conditioning on a single connection in terms of network bandwidth can be stated as:

$$OH = (\lceil \frac{PS}{SS} \rceil - 1)(HDR_{L2} + HDR_{L3} + HDR_{L4})$$

where OH is the introduced overhead in terms of bytes, PS is the original TCP payload size, SS is the target RIPPS slice size, and HDR_{L2}, HDR_{L3}, HDR_{L4} are the Ethernet (layer 2), IP (layer 3) and TCP (layer 4) header sizes. From this formula, a single full-size data packet (1514 bytes) would create an additional 1296 bytes of downstream network load, a near doubling of network load in terms of bandwidth. Similarly, a nearly identical reverse flow of packets (ACKs) would be generated (restricted to a minimum of 64 bytes) creating 1536 bytes of data on the local network.

At first glance, such overhead would appear to reduce RIPPS to a non-starter. However, the actual impact of RIPPS is mitigated through several factors. First, the two tiered approach significantly limits the impact of RIPPS. In the quick assessment stage, only a single large packet per host per monitoring period is conditioned for RIPPS. The net overhead is roughly proportional to a single retransmission of a lost packet on the upstream and downstream internal network links.

Second, RIPPS traffic is shaped using a relatively small shaping delay coupled with a inter-packet slicing delay to spread out the sliced traffic. Hence, the impact of RIPPS does not occur in an immediate burst. Third, RIPPS need only gather sufficient data to provide an appropriate confidence in the reliability of the data. From the earlier data, extracting a solid separation (95%+ confidence interval) between draft-n (the current fastest wireless) and wired hosts can easily be achieved in roughly 400 packets of data. At full incoming data packet sizes, only 17 full-sized data packets would be required to satisfy this level (roughly 26k of data). Hence, one could effectively cap the maximum impact of monitoring a single host at 22k downstream, 26k upstream on the local LAN.

Hosts	**Stage 1 Pkts**	**Stage 2 Points**	**Slice Size**	**Mon Period**	**Overhead (Mb)**
100	1	400	60	1 min	1.26
200	1	400	60	1 min	2.53
400	1	400	60	1 min	5.05
800	1	400	60	1 min	10.11
1600	1	400	60	1 min	20.21
2400	1	400	60	1 min	30.33
3200	1	400	60	1 min	40.44

Table VIII. Overhead of a potential RIPPS deployment

Fourth, the impact of RIPPS is not extended beyond the local network itself. RIPPS automatically "gobbles" the stimulated TCP Fast Re-Transmit packets, leaving the net impact external to the network of RIPPS nearly zero (beyond the slight initial throttling). Once the host leaves RIPPS monitoring, normal operations commence without any performance impact by RIPPS. Furthermore, critical infrastructure hosts can easily be exempted from RIPPS in order to not degrade performance. For instance, an environment whereby external Internet visibility is placed in the DMZ outside of local firewalled network, RIPPS would not degrade such services.

Consider a simple RIPPS deployment applied at a university (enterprise-wide) scale. For simplicity, consider only a single RIPPS monitoring station at the gateway to the Internet (i.e. co-located with the firewall/IDS). A Gigabit Ethernet backbone is shared by all hosts that may branch out to 100 Mb/s or Gigabit Ethernet. Table VIII shows the overhead of RIPPS in the upstream case with varying number of hosts, re-validating each host each minute (assume hosts are always active to the Internet), and a payload slice size of 60 bytes. For the purposes of allowing noise via transient events (congestion, computation, application), we assume the probability of a host transitioning to the aggressive assessment stage is 1%.

We note that the overhead for 2400 hosts monitored once per minute (typically with only a single fully sliced packet) is 30.33 Mb (3.79 MB). Depending on how the hosts are monitored, the overhead can be spaced out over the course of the monitoring period (one minute) leaving a net overhead of approximately 0.5% on a Fast Ethernet link. Despite that fact that the above table takes an extremely pessimistic assessment (all hosts are active all of the time), the actual overhead of RIPPS is quite reasonable, even on Fast Ethernet.

While not noted in the above scenario, the likelihood of transitioning to the aggressive stage is likely to occur for hosts in a batch-wise sense (i.e. the same link serving those hosts becomes congested). Although this is a cause for minor concern, the severity of this problem can be mitigated by the placement of the RIPPS device itself and by limits on the number of simultaneous aggressive assessments. Provided that the RIPPS device can observe the congestion (i.e. congestion crosses the RIPPS monitoring point), detection is only affected.

In practice, such a large enterprise would likey have RIPPS monitoring points extended closer to the hosts themselves rather than using only a central monitoring point in such a larger environment. For instance, consider a company campus with a wireless short haul between buildings. In such a case, a RIPPS monitoring device would be placed at the edge of the Internet in addition to one on the far end of the wireless short haul. While near-host placement in the switch as proposed by Bro-LAN [Weaver et al. 2006] is not necessary, RIPPS would ideally be placed next to each router in the enterprise network.

5.3 Weaknesses

The nature of RIPPS results in the ability of a host to detect when it is being monitored. This could be accomplished by a simple analysis of incoming traffic, which would show an abundance of very small packets, a clear indication of RIPPS activity. In the best-case-scenario, RIPPS would act as a deterrent to would-be attackers. However, an attacker could potentially attempt to modify its behavior in an attempt to avoid detection. Two counter-attacks which could be implemented by an intruder are wired system spoofing and a proxy attack.

5.3.1 *Wired System Spoofing.* Wired system spoofing is method whereby an unauthorized wireless host would appear as a wired system to the RIPPS monitoring system. If done successfully, an intruder could stealthily compromise network resources as if a detection system was not present.

Analysis of this problem is simplified by first looking at the unrealistic reverse situation of a wired host attempting to mimic a wireless one. To accomplish this, the wired host would add a delay to its acknowledgment of data received, increasing its measured LRTT values. In addition, the delay would be varied to create an increase in the standard deviation of the delay time. RIPPS could not distinguish the delay from the total LRTT, allowing the wired host to successfully take on the persona of wireless transmission.

The key to this mimicry is the ability of the wired host to delay TCP acknowledgment responses. Returning to the original problem, it is clear that in order for a wireless host to pass as being wired it must have the ability to act in the opposite manner, meaning it must decrease the delay of the TCP acknowledgments, as well as the variance of the delay. A wireless host would be unable to decrease LRTT times by modifying wireless protocols or the nature of the physical medium. Therefore, the host would need to resort to opportunistically sending ACKs in an attempt to reduce the LRTT values observed by the RIPPS monitor. The potential for a mimicry attack could be made extremely difficult to overcome by varying the shaping delay, slice size, and/or slice ordering (i.e. a checkerboard-like approach).

5.3.2 *Proxy Attack.* A proxy attack involves the use of a system, such as a laptop computer, which can be connected through a wired port and double as a wireless access point. In this attack, the laptop computer would act as a proxy for all communication between an unauthorized wireless source and associated destination systems. It is important to note that this scenario of an explicit proxy is distinct from most connection sharing methods (Mac OS X, Windows) that simply provide NAT-like services across the private network.

In this scenario it would appear that all communication originates, or terminates, at the proxy system, as this system is the actual endpoint for communication. The proxy is responsible for sending ACKs, meaning the wireless medium is not involved in the communication at all. The wireless host obtains data by establishing an end-to-end connection directly with the proxy. Therefore, direct communication to, or from, the wireless host will not pass through a RIPPS monitor, successfully evading detection.

The RIPPS system identifies wireless media involved directly in end-to-end communication which passes through a RIPPS monitoring system. In this case, the wireless medium is outside the end-to-end connection, and is therefore outside the scope of the problem which RIPPS solves. A solution for this problem could be drawn from proxy system detection techniques.

6. SUMMARY

Wireless connectivity point detection is an important capability for the security of computer networks. The RIPPS system performs network traffic conditioning to quickly and transparently create modified TCP packets critical for fast and accurate medium classification. RIPPS can potentially identify the existence of wireless connectivity on the order of milliseconds, eliminating the threat of extremely quick intrusions on the network. The efficient identification process results in a minimal amount of overhead on the network.

REFERENCES

Adya, A., Bahl, P., Chandra, R., and Qiu, L. 2004. Architecture and techniques for diagnosing faults in IEEE 802.11 infrastructure networks. In *Proceedings of ACM MobiCom*. 30–44.

Bellovin, S. M. 2002. A technique for counting NATted hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*. 267–272.

Beverly, R. 2004. A robust classifier for passive TCP/IP fingerprinting. In *Proceedings of Passive and Active Network Measurement, 5th International Workshop*. 158–167.

Beyah, R., Kangude, S., Yu, G., Strickland, B., and Copeland, J. 2004. Rogue access point detection using temporal traffic characteristics. In *Proceedings of IEEE GLOBECOM*. 2271–2275.

Cheng, L. and Marsic, I. 2001. Fuzzy reasoning for wireless awareness. *International Journal of Wireless Information Networks 8,* 1, 15–26.

Chirumamilla, M. K. and Ramamurthy, B. 2003. Agent based intrusion detection and response system for wireless lans. In *Proceedings of IEEE International Conference on Communications*. Vol. 1. 492–496.

Deraison, R. and Gula, R. 2003. Using nessus to detect wireless acccess points. Tenable Network Security. http://www.tenablesecurity.com/papers.html.

Guo, F. and Chiueh, T. 2006. Sequence number-based mac address spoof detection. *EURASIP Journal on Wireless Communications and Networking*.

Handley, M., Paxson, V., and Kreibich, C. 2001. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proc. of USENIX Security Symposium*.

Henning, R. R. 2003. Vulnerability assessment in wireless networks. In *Symposium on Applications and the Internet Workshops*. 358–362.

Karn, P. and Partridge, C. 1991. Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems 9,* 4, 364–373.

Mano, C. 2006. Defending against malicious rogue system threats. Ph.D. thesis, University of Notre Dame.

Savage, S. 1999. Sting: A TCP-based network measurement tool. In *USENIX Symposium on Internet Technologies and Systems*.

Weaver, N., Paxson, V., and Sommer, R. 2006. Work in progress: Bro-LAN pervasive network inspection and control for LAN traffic. In *Workshop on Enterprise Network Security*.

Wei, W., Suh, K., Gu, Y., Wang, B., and Kurose, J. 2006. Passive online rogue access point detection using sequential hypothesis testing with tcp ack-pairs. UMass CMPSCI Technical Report 2006-60.

Wei, W., Wang, B., Zhg, C., Kurose, J., and Towsley, D. 2005. Classification of access network types: Ethernet, Wireless LAN, ADSL, Cable Modem or Dialup? In *Proceedings of IEEE INFOCOM*. 1060–1071.