MAP: A Scalable Measurement Infrastructure for Securing 802.11 Wireless Networks

Yong Sheng[1], Keren Tan[1], Udayan Deshpande[1], Bennet Vance[1], Hongda Yin[3], Chris McDonald[1], Tristan Henderson[2], Guanling Chen[3], David Kotz[1], Andrew Campbell[1], Joshua Wright[4]

[1]Institute for Security Technology Studies, Dartmouth College;
[2]School of Computer Science, University of St. Andrews;
[3]Department of Computer Science, University of Massachusetts Lowell; [4]Aruba Networks

***Abstract*— The shared medium of 802.11 wireless networks means that they are susceptible to many MAC-layer attacks, such as frame spoofing, denial of service, and greedy misbehavior. To detect such attacks it is necessary to monitor the wireless channel itself and examine the timing and content of the 802.11 frames.**

Wireless "sniffing" is challenging: sniffers may not hear every frame, nearby sniffers may hear identical frames, sniffers' clocks may be poorly synchronized, or a sniffer may have its radio listening to a different channel than that on which an attack is occurring. In addition, sniffing a large-scale wireless network at a reasonable cost is also a challenge.

We present MAP (Measure, Analyze, Protect), a scalable architecture for extensive wireless measurement. MAP features dedicated sniffers, near-realtime multi-sniffer traffic merging, coordinated channel sampling, full stream capturing with feature extraction, and analysis-driven capture "refocusing." Using a deployment of 20 sniffers placed over a three-floor building, we demonstrate how these features help to improve detection accuracy in realistic attack scenarios. Performance evaluation shows that MAP is both effective and efficient at scale.

I. Introduction

The shared medium used in IEEE 802.11 wireless networks leads to many security risks. Although much research has focused on attackers that eavesdrop on or obtain unauthorized access to a closed network [1, for example], malicious wireless clients can also deny service to [2], or steal bandwidth from legitimate users, by sending forged or manipulated 802.11 frames [3]. As these wireless networks become used for mission-critical applications, in particular time-sensitive applications such as voice and video, the need to detect these attacks and protect legitimate users becomes even more important.

Detecting 802.11 attacks requires measurement at the MAC layer, which in turn requires radios that can hear the management and control frames that govern the MAC protocol. Deploying measurement software within wireless access points (APs) is difficult because most such devices are closed "black boxes," are typically locked to a specific channel (whereas attacks may be found on other channels), and have processors that are too limited to capture or analyze all traffic while simultaneously performing their primary function. Moreover, APs may not hear attackers located near the edge of their coverage area, even though those attacks may affect clients within range of the AP. Although some vendors' APs do periodically switch channels to listen for attacks on those channels, the small amount of time spent sniffing limits their ability to catch all but the most aggressive attackers.

Deploying dedicated wireless measurement devices, or "Air Monitors" (AMs), allows network administrators to listen for attacks in a broader variety of locations and on more channels than is possible with AP-based approaches. It is feasible to cover a large enterprise with dedicated AMs, which could be a cheap $200 embedded device or even a $30 USB dongle attached to existing office PCs, as proposed by DAIR [4].

It remains, however, a significant challenge to build a scalable system for monitoring a wireless network using distributed sniffers. The unlicensed bands used for 802.11 networks have multiple channels, and a sniffer that is listening to the wrong channel may miss an attack. In addition, nearby sniffers may hear the same frames, which may also cause problems in detection: attacks may be detected multiple times, or legitimate traffic may be classified as attacks. Finally, for many attacks it is important to capture a complete, coherent stream of frames to be able to detect and distinguish the attack.

In this paper, we describe "MAP" (Measure, Analyze, Protect): a scalable system for extensive monitoring and analysis of our campus wireless network. The MAP architecture includes a distributed set of AMs that constantly monitor wireless frames in their range. The AMs extract a small number of features from the overheard frames and send them to a *merger*, which reconstructs a stream of frame features with duplicates removed and timestamps appropriately aligned. Multiple analysis *detectors* subscribe to this stream to identify possible threats, attacks, and performance anomalies. Detectors then send alerts to a *protection engine*, which is capable of blacklisting offending stations by reconfiguring the wireless network firewall and displaying the analysis results to network administrators via a web interface. The MAP system includes a feedback loop, in which detectors may request that the measurement system on the AMs *refocus* their efforts to better obtain information of interest to the detectors; to do so, detectors send a request to the *controller* that guides the AMs' sniffing priorities.

We evaluate the MAP system with a full-scale deployment in the Sudikoff Lab for Computer Science at Dartmouth College, a medium-sized ($\approx 1,600m^2$) three-floor building with offices, classrooms, and labs. We used 20 AMs to cover

Unpublished manuscript, 2008

the building; the production 802.11a/b/g network uses 19 APs to provide wireless service to the same space.

The MAP system makes four main contributions. First, it includes novel channel-sampling strategies that adaptively gather more of the "interesting" frames from all available channels; second, a new compressed format for conveying relevant frame features; third, real-time merging of frame captures from multiple sniffers to provide a coherent picture of the traffic; and finally we demonstrate performance results that indicate that MAP should scale to enterprise deployments.

In the next section, we describe a class of attacks that deny service, or reduce quality of service, to wireless clients, and describe related work. Then we outline our approach in Section III, describing many of the design decisions that led to the MAP approach. We present (Section IV) and discuss (Section V) our experiments and results before concluding.

II. Threat Models and Related work

MAP is motivated by the measurement needs for detecting common 802.11 MAC layer attacks. In this section, we describe the threat models of 802.11 networks, followed by related work in wireless measurement and security.

A. 802.11 Threat Models

MAP focuses on active 802.11 MAC-layer attacks, including Denial of Service (DoS) attacks that disrupt network services to legitimate clients, and Reduction of Quality (RoQ) attacks that degrade quality of service available to other clients of the network. For example, an attacker may not have valid network credentials, but he may spoof management frames, such as deauthentication or disassociation, to knock clients off APs [5], [6]. Alternatively, an attacker may pretend to be a legitimate AP to perform man-in-the-middle attacks or denial of service [7]. In both of these cases, the attacker is an outsider but exploits the fact that 802.11 management and control frames are not protected (and so easy to forge), and the lack of mutual authentication between clients and APs. On the other hand, insiders with legitimate network access may attack the network, for instance greedy users attempting to gain a greater than fair bandwidth share by manipulating the MAC protocol [3].

In this paper, we concentrate on active attacks, where attackers inject traffic to cause damage. Attackers, however, may transmit frames to any channel, and from any location in the area covered by AMs. We also assume that the MAP system itself is appropriately protected by typical security measures. An attacker who knows our measurement strategy could evade or confuse the MAP detectors, but we leave this challenge for future work, possibly by adding non-determinstic features to our adaptive sampling.

B. Wireless Measurement

Few large-scale 802.11 measurement studies have attempted to capture wireless frames off the air; most prior work captures packet traces from the wired side of the AP [8], [9]. Although a few papers characterize traffic at meetings and conferences [10], [11], or describe tools for capturing and analyzing wireless frames [12], [13], [14], none consider channel sampling, or security.

A few recent papers describe offline tools to capture and merge wireless frames from AMs around a building. Yeo et al. [15] use beacon frames captured at multiple sniffers to create a linear equation to synchronize sniffer clocks and subsequently remove duplicate frames. This system is only tested on offline traces and it is unclear whether it will work in an online measurement environment. In addition, the sniffers in this system are all set to the same channel, which leads to a many frames being redundantly heard at multiple sniffers. Our merger works with online sniffers, and with channel-hopping sniffers that have fewer frames in common. *Wit* [16] includes a similar merging mechanism to combine multiple traces and an inference engine to determine missing frames, but it again works offline. *Jigsaw* [17], [18] uses multiple AMs to collect and merge 802.11 traces. The focus of this work is to detect performance artifacts and network inefficiencies by reconstructing transport and link-layer conversations. The authors use common frames captured by different AMs for time synchronization. Jigsaw's suitability for *online* analysis is unclear. MAP, on the other hand, specifically targets online frame capture, online merging, and analysis for attack detection. Our own earlier work [19] focused on the challenge of sampling traffic from many channels, and merging frames from many AMs; in this paper we look more deeply at these issues through a large-scale experimental deployment to evaluate our ideas more thoroughly, and consider the scalability of the entire MAP system.

C. Wireless Security

802.11 security remains a challenge, because there are many vulnerabilities in the protocol and its implementations [2], [20], [21]. The new IEEE 802.11i standard helps to secure the data being transmitted [22] but does not protect against MAC-layer attacks that deny or degrade service.

The *DOMINO* system [3] is designed to detect several MAC-layer greedy behaviors where an attacker deliberately misuses the MAC protocol to gain more bandwidth, rather than to be purely malicious against other stations. Only a single-sniffer scenario is used for detection, and multiple-sniffer scenarios are not considered. Moreover, they only evaluate a limited implementation. Radosovac et al. [23] present a theoretical framework for measuring and detecting misbehaving 802.11 clients; this framework requires perfect measurement of every 802.11 frame, which is impractical in a channel-sampling scenario. Several commercial products ([24], [25], for example) use multiple sniffers, although they do not provide the ability to merge streams from these sniffers; the sniffers are typically used to send high-level summary statistics, rather than packet traces; and the channel-hopping configuration is not customizable, with sniffers simply spending an equal amount of time on each channel, which we have demonstrated to be less effective.

The DAIR system [4] uses USB 802.11 dongles to turn desktops in an enterprise network into AMs. In a less controlled environment, however, such as a university campus,

there may be a larger variety of client types, a far less dense deployment of desktop computers, and the network administrators may be unable to force users to install sniffers on their desktops. In these environments, and even in many enterprise environments, dedicated sniffers will be necessary.

The key difference between MAP and DAIR is that sniffers in DAIR send only summary reports to a centralized "inference engine", whereas MAP shows that it is feasible to capture, merge, and analyze all frames from all sniffers on a per-frame basis. This fine-granularity measurement puts MAP in a better position to detect a variety of attacks. For example, our work on signal-strength-based spoof detection [26] demonstrates that MAP's per-frame merging helps to achieve a higher rate of detection (97.8%), compared with merging local statistics (89.6%) from multiple AMs, at the same false-positive rate. For another example, we assume that an attacker A sends an erroneous response frame F_A following a request frame F_V from the victim V. It is possible that F_A can only be received by one sniffer, and F_V by another, due to locations of A, V and the sniffers. This attack can be simply detected by the signature "F_V *is followed by* F_A", if we have a per-frame merger. Neither sniffer, however, can raise an alert based on its local observations. In addition, DAIR assigns each sniffer to a specific channel. Covering all channels requires a dense AM deployment which is far more expensive than the deployment of MAP's channel-hopping AMs.

III. MAP ARCHITECTURE

We now describe the architecture of the MAP system, shown in Figure 1. Briefly, the AMs capture wireless frames, extracting and forwarding the desired frame features to the merger, which creates a unified stream on a coherent timeline. The analysis engine includes plug-in detectors that analyze the traffic, producing alerts to the protection system and feedback to the measurement system. The controller coordinates the AMs and modifies their behavior according to feedback from the analysis engine. The protection engine presents alerts to the system administrator and (in future work) takes autonomous action to protect the network from the attacker. Due to space limitations, we present only a brief summary of each component here.

A. Air Monitors (and controller)

AMs are wireless sniffers that capture traffic from the 802.11 spectrum. In our deployment, we use dual-radio Aruba AP70 APs loaded with OpenWrt Linux[1] although an AM could be built from any 802.11 NIC that is capable of programmatic channel changes.

Any installation will need to deploy several AMs throughout the relevant area. Since attacks may occur on any channel, even those not in use by the production network, the AMs need to scan all possible channels. For this reason, and to ensure that we hear attackers using low-power settings to quietly attack nearby clients, we may need to deploy more AMs than APs. Some redundant capture will result from overlapping AM coverage, so MAP coordinates the AMs' schedules to minimize channel overlap and the MAP merger removes duplicate frames.

[1] http://www.arubanetworks.com, http://www.openwrt.org

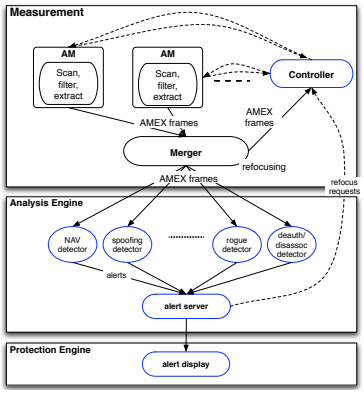

Fig. 1. The MAP architecture; dashed lines are control streams and bold lines represent data streams.

AM channel sampling: MAP needs to monitor all 802.11 channels (as many as 14 channels for 802.11b/g, and 23 channels for 802.11a), as there may be attacks on any of them, even if the network infrastructure is using only a few channels. With current technology, it is not feasible to monitor all channels all the time, at one location. Our AMs monitor multiple channels using two radios, by periodically assigning each radio to each channel. We call this technique "channel sampling", as it collects only a sample of the frames passing through all the channels. In its simplest form, channel sampling involves setting the radio to each channel in the wireless network, in a predetermined order, and spending equal amounts of time on each.

For wireless intrusion detection, and other applications, equal-time sampling may be inappropriate. In this paper, we consider two other sampling strategies, which adaptively spend more time on channels considered to be more "important" in some respect. We assume that each strategy rotates through all channels, once per sampling cycle, and define a sampling strategy in terms the amount of time spent on each channel, and how that time is adjusted in response to current conditions.

The "proportional" strategy observes the number of frames per second on each channel, and uses the proportion of traffic on each channel to determine the proportion of the next scanning cycle to spend on each channel. We explore the basic concepts of the above strategy in an earlier workshop paper [19]. In this paper, we evaluate a more robust implementation on a larger deployment (20 sniffers) and in the context of the full MAP system, exploring the effects of sampling on a wider variety of attacks and considering performance in regards to system scalability.

We also consider *coordinated sampling*, in which the AMs' individual sampling schedules are coordinated by a central controller. Each AM chooses the amount of time to sample each channel, as before, in proportion to the amount of traffic on each channel. Based on statistics provided by the AMs, the

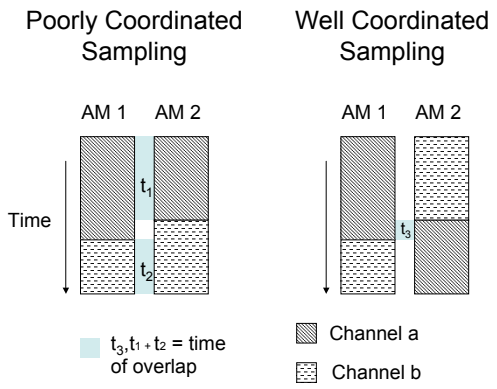

Fig. 2. Effect of channel scheduling on overlap between neighboring AMs

central controller rotates the AMs' schedule of channels within each cycle, however, to reduce the amount of time neighboring AMs spend on the same channel, and thus reducing the amount of redundant capture. Figure 2 shows a fictitious example of poor channel scheduling where the two neighboring AMs spend a lot of time on the same channel, and how changing the order of the sampled channels reduces overlap.

We expect that reducing the amount of time that neighbors spend on the same channel simultaneously will result in fewer redundant frames being captured. AMs that use coordinated sampling should use their resources more efficiently, resulting in a greater total capture. We describe our coordination algorithm, and explore its value in small controlled studies, in an earlier paper [27].

Refocusing: The MAP measurement system is like a telescope, focused on a region of channel-space-time. MAP allows analysis components to dynamically *refocus* the measurement system after observing anomalous behavior, by gathering more frames from a client, AP, or region, or by extending the set of features collected about the traffic of interest. In case of an ongoing attack, the higher-fidelity stream of frames may allow MAP to confirm the attack or locate the attacker.

Our current implementation can focus more attention on a given MAC address, by asking the relevant AMs to spend more time on the channel where that MAC was recently observed.

For example, if we wish to spend more time capturing frames being sent from MAC address aa:bb:cc:dd:ee:ff to MAC address 11:bb:33:dd:55:ff, we need to direct the AMs to spend more time capturing traffic on the channels that observe these MAC addresses. The MAP system allows such *refocusing requests* to take the form of predicates like "src == aa:bb:cc:dd:ee:ff && dst == 11:bb:33:dd:55:ff". These predicates are sent to the relevant AMs, which maintain per-channel-counters for the number of frames that match these predicates. These counters are used to determine the proportion of time that an AM spends on each channel. We explore the potential for refocusing in another paper [28].

AM feature extraction: To reduce the volume of forwarded traffic while retaining the relevant information from each individual frame, AMs forward information in a compressed frame format that we call *AMEX* (for *AM EXtractor*). This format includes only the *interesting features* of each frame and packs the features for several frames into each UDP datagram sent from the AM to the merger.

Our AMEX encoding scheme allows us to redefine the set of interesting features dynamically to adapt to changing conditions. Thus, each AMEX frame includes a header (with a bitmap that indicates which AMEX features are present), data specific to the AM (e.g., the timestamp and signal strength of the received 802.11 frame), and the selected features from the 802.11 header (such as the source MAC address) or PHY-layer information provided by the 802.11 driver (such as the rate).

When multiple AMs hear the same 802.11 frame and forward it to the merger in AMEX format, the feature blocks are all identical, but the AM-specific data in these frames may vary in ways important to the detectors. Thus, the merger combines information from redundant frames, using a variant AMEX format to retain one copy of the frame header data and all the AM-specific data from all AMs.

This AMEX approach dramatically reduces bandwidth demands for communicating frames from the AMs to the merger. Under typical conditions, AMEX reduced AM-merger traffic by 45.7% compared to full 802.11 frames, even when we pack only one AMEX frame per UDP datagram, and 66.2% when we pack multiple frames per packet. AMEX thus achieved good compression with a simple frame format and reasonable processing costs.

B. Merging

Our merger combines all AM traces to provide a coherent, complete view of the wireless traffic. This combined view is important because no single AM may be able to capture all of the frames between a client and AP of interest, or between an attacker and all of the affected victims. On the other hand, multiple AMs may redundantly capture the same frames. There are two challenges in merging: synchronization of the AMs' clocks and identification and removal of redundantly-captured frames.

Synchronization and clock correction: Since NTP can provide only coarse-grained clock synchronization, in contrast to the microsecond-scale synchronization required by the merger, the merger tracks a *clock correction* variable for each AM, adding it to the timestamp of each frame from that AM. The merger updates these corrections whenever it processes a beacon frame observed by multiple AMs. Briefly, we adjust the clock correction for those AMs so that the imputed timestamp for that beacon will be the same on all AMs.

Identification and removal of duplicates: As frames from different AMs do not necessarily arrive at the merger in corrected-timestamp order, the merger must buffer incoming frames for a period of time so that it can reorder them and identify duplicates. Briefly, we maintain a queue of recent frames for each AM, and on the arrival of each new frame we look for duplicate frames within these queues. For speed, we compare the frame check sequence (FCS), rather than frame contents; there is only a tiny chance that two unequal frames arrive at the same time and have the same FCS. Eventually,

the merger's output component draws the oldest entry from the heads of the AM queues. That entry is scheduled for output when its age exceeds a defined threshold, so that it is extremely unlikely that a preceding frame will arrive from another AM.

C. Analysis engine

The MAP analysis engine is configurable with a set of independent plug-in *detectors*. This structure allows easy development of new detectors, coded in any language and for any platform suitable for the task. Each detector can be run on any available host, distributing the analysis load and enhancing scalability. The MAP configuration simply informs the detector of the location of the *merger* (to which the detector subscribes to receive a stream of relevant AMEX-format feature frames) and the location of the *alert server* (to which the detector publishes any alerts). Our merger will soon support *filtered subscriptions* so that each component may receive only the feature frames it needs (with filters based on MAC address, channel, or type).

Indeed, our framework allows one detector, which may be an anomaly detector, to launch a new instance of another detector, which may be a specific signature detector, and to refocus the measurement system so the new detector can examine the suspicious region (or client) more closely. We will use this feature in an anomaly detector (under development).

Our refocusing mechanism is also important for detection. Detectors are not simply "clients" of the measurement system, but instead they can adjust measurement activity according to the results of analysis. The MAP system thus enables active as well as passive analysis. The current MAP implementation contains four refocusing-equipped detectors: deauth/disassoc detector, spoofing detector [26], NAV detector and rogue detector [29]. Due to space limitations, we only describe how refocusing works in the deauth/disassoc detector.

Deauth/disassoc detector: Deauthentication(deauth) and disassciation(disassoc) frames are two types of 802.11 management frames that can terminate 802.11 authentication and association procedures respectively. By continuously sending spoofed deauth/disassoc frames whose source MAC address is the AP and the destination MAC address is the target, the attacker keeps the target trapped in the authentication/association procedures and thus prevents the wireless connectivity between the target and the AP. Such an attack can be a flood (¿ 10fps) or tickle (otherwise), depends on the purpose of the attacker, and the configuration of the AP and station as well.

The refocusing capability of the deauth/disassoc detector works as follows. The detector monitors the frame type of each frame output by the merger. Once it finds a suspicious deauth/disassoc frame, the detector sends a refocusing request to the controller. The controller assigns a ticket to this request and sends it back to the detector, who saves the ticket for future communications with the controller. The detector also builds a profile which includes source and destination MAC address, a counter, timestamps for each suspicious frame, the refocusing ticket and two timers, t_r and t_e, where t_r records the time when the refocusing request is over and t_e records the expiration time for monitoring the situation. Whenever the refocusing timer t_r expires, and t_e has not expired, the detector sends a *renewal* request to the controller. When a suspicious frame arrives in with the same addresses, t_e is extended and the counter is incremented. If this counter exceeds a predefined threshold, the detector generates an alert message (containing the attack start time) to the alert server. If no more suspicious frames are heard when t_e expires, the detector sends an alert message (containing the attack start time, end time, and other statistics) to the alert server. The detector flags this attack as "true" if the number of suspicious frames was more than a predefined threshold; otherwise it flags this attack as a false attack. In either case, the detector sends a *cancellation* request to the controller. In the following section, we show that refocusing improves detection effectiveness.

IV. Experimental Evaluation

The MAP system is designed to monitor and (ultimately) protect large-scale wireless infrastructure networks. Here, we evaluate the performance of detecting attacks while sampling over all the channels, using different sampling strategies and the analysis-driven refocusing mechanism. We also evaluate the capability of our current MAP implementation and study its ability to scale. To do so, we deployed 20 Aruba AP70 AMs across the CS department building at Dartmouth, and ran experiments involving live captures and attacks on our production 802.11 network.

A. Experimental Setup

We deployed 20 Aruba AP70 AMs among and between production 802.11a/b/g Aruba APs, as shown in Figure 3. The AP70 has a 266MHz MIPS IDT32434 CPU, 32MB DRAM, two Atheros AR5212 802.11a/b/g NICs (network interface controllers), and two Ethernet NICs. We installed OpenWRT Linux (Kamikaze branch, r5494) and Madwifi (v0.9.2) on each, and a copy of *dingo*, our channel sampling software that sniffs through libpcap (v0.9.5). Dingo can sniff on both of the NICs, but in our experiments it sniffed only one radio and only 802.11b/g, as our 802.11a network is in limited use. We connected all the AMs to the merger through our wired building network, which is switched 100Mbps Ethernet, without routers in the middle.

Our merger, controller and analysis engine are running as user-level processes on one of two Linux (Fedora Core 6) servers (2x3GHz Intel Xeon CPUs, 4GB RAM, 3.2TB RAID storage). Note that the merger and any post-merger MAP components can run either on the same server or different ones. We configured them to run on the same server in most of our experiments.

For the purpose of evaluation and future offline analysis, we collected and sanitized traces while the system was running online. With a tcpdump process sniffing on the merger input UDP port, the interleaved AMEX packets from all AMs were captured as an aggregated pre-merger trace, retaining the important ordering and timing information. We also captured a post-merger trace by sniffing on one of merger's output ports.

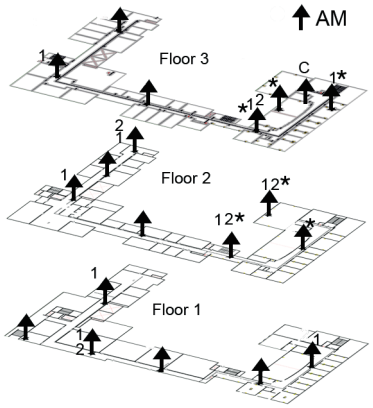

Fig. 3. Testbed deployment in our CS department building. The 19 Aruba AP52 APs (not shown) provide 802.11a/b/g service to over 80 faculty, students and staff members in about 1,600 square meters of usable space. We deployed 20 Aruba AP70 air monitors (arrows) throughout. Some AMs are marked "*","1", or "2" since we separate them into groups in some experiments.

We later fed both the traces into MATLAB for quantitative studies. We also added instrumentation to all physical devices and logical components in MAP, to measure the workload, CPU load, memory, bandwidth, and other resources, reporting (via syslog) periodically to a central server for later analysis.

B. Effectiveness of Attack Detection

We first evaluate MAP's attack detection effectiveness, under various channel sampling strategies and attack scenarios. We use two success metrics: a) the ability to capture attack frames, measured by the ratio of attack frames captured by MAP to the total attack frames injected, and b) the detection rate, measured as the fraction of successfully detected attacks to all launched attacks.

We deployed 18 AMs at six locations (marked "*" in Figure 3), three at each location, 0.5m apart. We divided AMs into three groups, each group having one representative at each of the six locations. We configured each group with different sampling strategies: 1) *proportional*, so each AM uses the proportional sampling strategy independently; 2) *coordinated*, like group 1 but under the coordination of a controller to reduce channel overlap; and 3) *refocusing*, like group 2 but allowing the detector to send refocus requests to the AMs through the controller. In addition, we set up a control AM (marked "C") 2m from the attacker, fixed on the attacker's channel without any channel sampling or refocusing.

For each group, as well as the control AM, we ran separate instances of the merger, controller, and detectors simultaneously. We set up the deauth/disassoc detector to report an alert when it captured at least 3 deauth frames from a given MAC address, among which any consecutive two must be within 6 seconds in time. For the *refocusing* group, we configured the detector to request refocusing on the MAC address at the moment it received the first deauth frame.

We set up an attacker (laptop) to launch deauth attacks, once per minute, against a pre-defined MAC address. We assume that the attacker may know the proportional sampling strategy, and thus may evade MAP detection by deliberately injecting heavy traffic to one channel and attacking another idle channel. We carried out two sets of experiments: in set *A* the attack was on the busiest channel (channel 11 in our case), and in set *B* the attack was on the idlest channel (channel 7). The control AM was always on channel 11.

We assume that the attack frames may arrive in a *flood* or a *trickle*, depending on the purpose of the attacker, and the configuration of the AP being spoofed. Thus we varied the injection rate of deauth frames at six different levels, as 0.5, 1, 2, 5, 10 and 50 frames per second (fps). At each given frame rate, we repeated the experiment 120 times, 40 seconds each time. The two sets of experiments lasted for 24 hours.

Channel sampling: Before we look at attack detection results, consider the long-term performance of channel sampling, for the *proportional* and *coordinated* AM groups. We plot the AMs' output and the merger's output on each channel on a log scale in Figure 4. The unit is FAMH, frames per AM per hour. On the horizontal axis the channels are arranged in descending order of FAMH captured by the *coordinated* group, on each channel. For each channel, the two bars represent the two sampling strategies. The dark portion of the bars represent the FAMH after merging the traces. We also plot the redundancy rate of the two groups as lines across the top. We observed the redundancy rate of the coordinated group was usually lower than the proportional group, except on the busiest channel (channel 11). The total frames captured by the two strategies were about the same, as was the overall redundancy rate. The lower redundancy rate on all channels for the coordinated group was somehow offset by its higher redundancy rate on the busiest channel.

Capturing attack frames: Figure 5 shows the percentage of captured attack frames under different frame injection rates; the attack occurred on channel 7 or channel 11. Because the control AM was close to the attacker and stayed on the attack channel, it captured almost 100% of the attack frames. The most notable feature of this graph is that the *coordinated* and *proportional* groups were poor at capturing attack frames when the attack occurred on a quiet channel (channel 7), because they focus attention on the busier channels. On the other hand, the *refocusing* group captured the most attack frames among the three channel-sampling groups in most cases, because its focus was determined by the early indications of attack on either channel. [The *coordinated* group slightly outperformed the *refocusing* group for a trickle attack on the busiest channel (0.5 fps, channel 11); we believe that this effect was due to the longer feedback loop of refocusing operation, and because coordination captured more redundant frames in this case.] The *coordinated* and *proportional* groups had similar capture performance.

Rate of detection: Since the detector does not need to capture all attack frames to raise an alert, the rate of attack detection (shown in Figure 6) for each group was much higher than the frame-capture rate. When the attacker targeted the busiest channel, all three groups detected 100% of attacks, even for the subtlest trickle attacks. When the attacker targeted

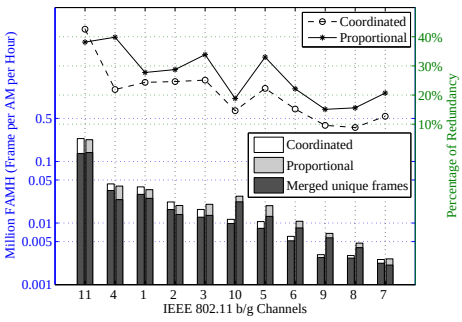

Fig. 4. Frames captured over channels

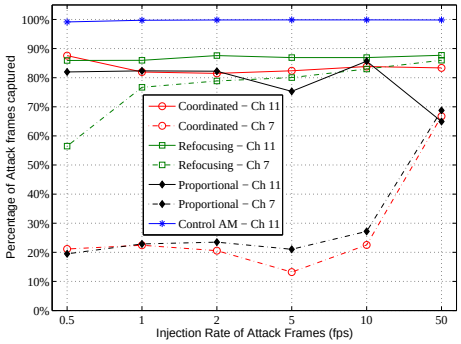

Fig. 5. Attack frame capturing percentage

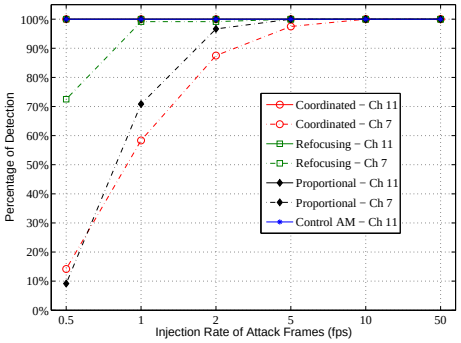

Fig. 6. Attack detection percentage

the idlest channel, however, the *proportional* and *coordinated* groups were poor attack detectors unless the attacker injected 2 fps or more. The *refocusing* group was consistently better than the other two groups. For the subtlest trickle attacks (0.5fps, channel 7), the *refocusing* group detected 72% attacks, which is 4.1 times higher than the *coordinated* group (14%), and 6.2 times than the *proportional* group (10%).

Based on the observations above, MAP successfully detected attacks while channel sampling. The analysis-driven refocusing was important for accurate detection.

C. Performance

Attack frames must be captured before they can be detected. Effective frame capture is thus critical to the security purpose of the MAP system.

Sniffing performance: We conducted a heavy traffic experiment on the testbed, when the 20 AMs were deployed as in Figure 3. We used 8 wireless Linux PCs as traffic generators scattered on the three floors (not shown). On each AM we measured the frames received by NIC, libpcap, and the *dingo* sniffer software. Our success metric for sniffing is the *frame drop rate*, defined as the fraction of frames not received by the sniffer, to the total frames that the NIC receives, in each cycle of channel sampling. Figure 7 shows the average drop rate across the sampling cycles for all the 20 AMs, under different input workload size (i.e., the frames received by AM's NIC), with the 20–80% error bar. The results show the drop rate was low ($< 5\%$) when the interface received less than 1,000 fps. But it increases almost linearly to 23%, 50% and 74%, at 2,000, 2,600 and 3,200 fps, respectively. With higher input rates, the drop rate remained constant around 70-90% till 5,000 fps. Our results also show that libpcap captured almost all frames ($> 99\%$), but it dropped frames due to the limit of its internal buffer, implying that dingo's ability to extract and forward features (on these AMs) was about 1,000 fps. This performance may be improved by increasing the buffer size in libpcap, and by tuning dingo.

Merging performance: Figure 8 shows the CPU load of the merger that we observed from our stress experiments, which is linear to the FPH metric of the system, i.e., the average number of frames captured by all AMs in an hour. Our merger only used one of the two processors in the server, so we measured CPU load of that processor only.) We found that the slope of the lines depended on the redundancy rate. By simple linear extrapolation, (shown as the the dotted lines) we expect the merger (while using 50% CPU) has a throughput of approximately 31, 32, 34, and 45 million FPH when the rate of redundancy is about 10%, 20%, 35%, and 50%, respectively.

a) AM-merger bandwidth: Bandwidth limitation is another major concern. The flow of AMEX frames from AMs to merger may cause congestion; since this traffic is carried by UDP there is some potential for loss. We disabled AMEX aggregation, and forced the AMs to send one UDP datagram per captured frame. Figure 9 shows that the drop rate remained relatively flat with no upward trend as the AMs' output rate increased. Thus, up to the merger's maximum observed output frame rate (28M FPH), we were well within the capabilities of UDP over switched 100Mb Ethernet.

The average AMEX/UDP datagram size is 148 bytes, including headers, when carrying the features from one captured frame. The maximum throughput of such a UDP stream is around 60M FPH. The AMEX aggregation feature, however, dramatically reduces the number of UDP datagrams by 95% (an average of about 21 frames per UDP datagram), while increasing the average datagram size to about 1500 bytes. At such a packet size, a UDP stream can easily achieve more than 90 Mbps throughput on a 100-Mb switched ethernet. This increases the theoretical bound to 560M FPH. Thus we do not expect bandwidth to be a bottleneck of MAP. At the same time, the average FAMH that we observed in experiments (under coordinated sampling) is about 0.4M FAMH. This analysis suggests that bandwidth will not be a bottleneck.

D. Scalability

Our experimental results demonstrate that MAP is efficient and effective for our building-wide deployment. We now consider scalability. Since a single merger clearly cannot scale to campus-wide deployments, we introduce the concept of *merging regions*. We assume that in any large-scale 802.11 infrastructure (such as the Dartmouth campus), the wireless traffic is concentrated in geographically distributed buildings, or groups of buildings. We envision that a MAP system groups all the AMs in each building into one merging region; each merging region includes one instance of the merger

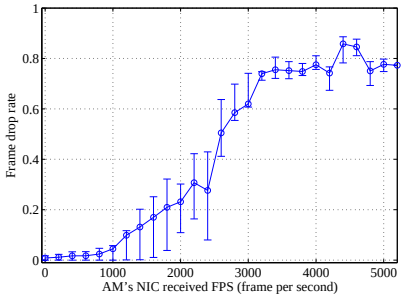

Fig. 7. Frame drop rate at AM sniffers.

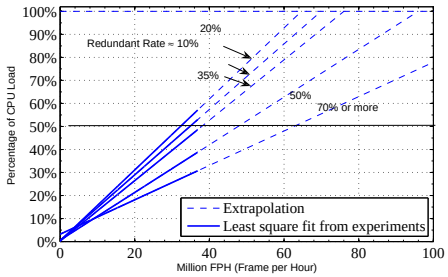

Fig. 8. CPU load of merging and extrapolation.

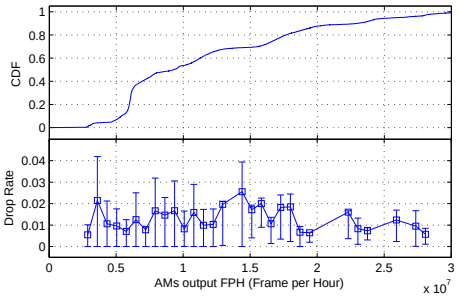

Fig. 9. Frame drop rate between AMs and merger

and analysis components. Since the coverage of two MAP merging regions may overlap, and some AMs can hear traffic in neighboring regions, we depend on the alert server to merge the alerts generated by the two regions. With careful selection of merging regions, including enough AM overlap to be sure to detect attacks on the boundaries, but not so much overlap so duplicate detection is difficult in the alert server, we believe that MAP can scale to campus-size deployments. (This duplicate-detection feature remains as future work, however.)

Of course, we would like each merging region to be as large as possible, to minimise the number of servers required. On the other hand, we are concerned with CPU load on the merger, which depends on the input workload size, i.e., the number of frames captured by AMs. In a 24-hour long experiment conducted on the testbed on a normal business day, with the 20 AMs sampling over the 11 802.11b/g channels using the *coordinated* strategy, the AMs captured 173M frames which were merged to create a stream of 87M unique frames. The redundancy rate (denoted by R) is 49.7%.

To explore the effects of AM deployment density, we chose a subset of 10 AMs (marked "1" in Figure 3), and a subset of 5 of those AMs (marked "2") to compare with the full 20. In order to ensure that all comparisons are based on the same traffic, and avoid temporal variations of wireless traffic, we compute our coverage metrics for the three groups from the offline traces collected in the experiment. The results (Figure 10) show a sparser AM deployment reduces redundancy rate, but does not change the average AM capturing rates significantly (about 0.4M FAMH for all the three configurations).

Table I uses these extrapolation results; provisioning the system to use half of our CPU power (leaving half to handle traffic bursts), we estimate the size of each merging area in number of AMs, the AM density, and the area covered.

This table provides a rough estimate of the size of the merging region possible while using the coordinated sampling strategies. We use one of the largest office buildings at Dartmouth— the Baker/Berry library— as an concrete example, in which 69 APs provide 802.11/a/b/g service to about 22,000 m^2 space. Table I indicates that we should be able to deploy MAP in several possible configurations at such a scale. Indeed, the current merger could be much faster, by tuning its performance or by using multithreading to take advantage of the multi-processor server.

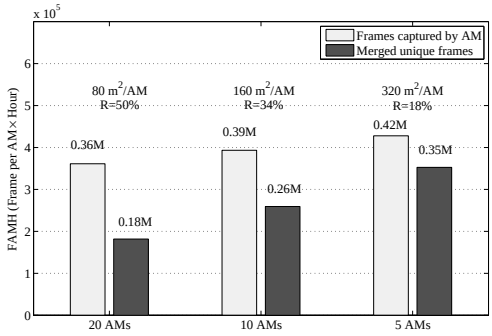

Fig. 10. Frame captured by AM under various densities of AM deployment. R denotes the redundancy rate.

Our current, untuned implementation of the the MAP system runs on a building-wide scale, and should achieve campus scale using multiple merging regions.

V. Discussion

We consider several issues and ideas for future work.

MAP assumes that the monitoring system is independent of the production network. This structure makes it easier to deploy, since the monitoring system does not depend on the vendor or architecture of the production network. In most deployments, however, the MAP system and production network will be operated by the same administrators, and so we may be able to leverage the production APs in some way. For example, underutilized APs could be tasked to act temporarily as AMs, or APs could be instrumented to report statistics or special events in the network (e.g., verifying the authenticity of Deauthentication messages).

The current MAP deployment is monitoring an unencrypted network. A network secured with WEP or WPA encryption will encrypt data between the clients and the AP at the MAC layer. In these schemes, however, the 802.11 header is left intact. Therefore, all our current detection techniques can continue to be used. To detect attacks involving higher protocol layers, we can integrate MAP with an intrusion-detection system running on the wired side of the APs.

Under the assumption that attacks are likely to be coupled with heavier traffic, the MAP system dynamically allocates more resources to busier channels. When an earlier sign of attack is detected, however, the analysis engine drives the

Desired AM density			
m^2/AM	80	160	320
AP:AM ratio	1:1	2:1	4:1
Coordinated sampling strategy			
Average FAMH	0.4M	0.4M	0.4M
Redundant Rate	50%	34%	18%
Merger Throughput FPH	47M	38M	32M
Max AMs	117	95	80
Area covered m^2	9,360	15,200	25,600

In the table above, the average FAMH and redundant rate are approximated from our coordinated channel sampling experiments in normal conditions in the CS department building, from Figure 10. The merger's throughput FPH is an extrapolated value at 50% CPU load, as in Figure 8.

TABLE I

MAXIMUM COVERAGE WITH MERGER AT 50% CPU POWER

MAP system to focus on the suspicious activities, through the refocusing mechanism. The results in Section IV-B suggest this strategy is successful, with appropriately implemented *analysis-driven refocusing* according to the attack models. But inappropriate refocusing requests may mislead AMs to the wrong channel(s), and risks missing true attacks. From the measurement point of view, the coordination of concurrent and possibly conflicting refocusing requests remains as a research challenge for future studies.

In the current MAP system, each detector in the analysis engine receives a full AMEX stream from the merger. On the other hand, not all detectors need all features in the AMEX stream. We plan to implement a *filtered subscription* so detectors can customize desired frame features.

VI. CONCLUSIONS

This paper introduces MAP, a system designed to capture wireless traffic over a broad area, merge frames captured by AMs with overlapping coverage areas, and detect MAC-layer attacks in real-time. We implemented the MAP system and deployed it throughout our Computer Science building. We measured the performance of MAP as it monitored daily traffic on our production network under several different scenarios, sometimes injecting traffic or attacks to measure MAP's performance.

We found that the MAP system performed well, that coordinated sampling was slightly more effective than our basic proportional sampling, that refocusing was able to improve the effectiveness of our attack detectors, and that the system was efficient. Extrapolating from our performance measurements, we expect that MAP will scale to large buildings and then (with additional servers) to a campus scale.

REFERENCES

[1] A. Bittau, M. Handley, and J. Lackey, "The final nail in WEP's coffin," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2006, pp. 386–400.

[2] J. Bellardo and S. Savage, "802.11 denial-of-service attacks: Real vulnerabilities and practical solutions," in *Proceedings of the 12th USENIX Security Symposium*, Washington, DC, Aug. 2003, pp. 15–28.

[3] M. Raya, J.-P. Hubaux, and I. Aad, "DOMINO: Detecting MAC layer greedy behavior in IEEE 802.11 hotspots," *IEEE Transactions on Mobile Computing*, vol. 5, no. 12, pp. 1691–1705, 2006.

[4] P. Bahl, R. Chandra, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill, "Enhancing the security of corporate Wi-Fi networks using DAIR," in *Proceedings of MobiSys 2006*, Uppsala, Sweden, June 2006, pp. 1–14.

[5] "file2air frame-generation tool," http://wirelessve.org/entries/show/WVE-2005-0059.

[6] "Void 11 deauthentication/disassociation attack tool," http://wirelessve.org/entries/show/WVE-2005-0052.

[7] "Fake AP," http://www.blackalchemy.to/project/fakeap/.

[8] D. Kotz and K. Essien, "Analysis of a campus-wide wireless network," *Wireless Networks*, vol. 11, pp. 115–133, 2005.

[9] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *Proceedings of MobiCom 2004*, Philadelphia, PA, Sept. 2004, pp. 187–201.

[10] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer, "Understanding congestion in IEEE 802.11b wireless networks," in *Proceedings of IMC 2005*, Berkeley, CA, Oct. 2005, pp. 279–292.

[11] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorian, "Measurement-based characterization of 802.11 in a hotspot setting," in *Proceedings of ACM SIGCOMM E-WIND workshop*, Philadelphia, PA, Aug. 2005.

[12] K. Jamieson, B. Hull, A. Miu, and H. Balakrishnan, "Understanding the Real-World performance of carrier sense," in *Proceedings of ACM SIGCOMM E-WIND workshop*, Philadelphia, PA, Aug. 2005.

[13] F. Li, M. Li, R. Lu, H. Wu, M. Claypool, and R. Kinicki, "Tools and techniques for measurement of IEEE 802.11 wireless networks," in *Proceedings of WiNMee 2006*, Boston, MA, Apr. 2006.

[14] A. Mishra, E. Rozner, S. Banerjee, and W. Arbaugh, "Exploiting partially overlapping channels in wireless networks: Turning a peril into an advantage," in *Proceedings of IMC 2005*, Berkeley, CA, Oct. 2005, pp. 311–316.

[15] J. Yeo, M. Youssef, T. Henderson, and A. Agrawala, "An accurate technique for measuring the wireless side of wireless networks," in *Proceedings of WitMeMo 2005*, Seattle, WA, June 2005, pp. 13–18.

[16] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Analyzing the MAC-level behavior of wireless networks in the wild," in *Proceedings of SIGCOMM 2006*, Pisa, Italy, Sept. 2006, pp. 75–86.

[17] Y.-C. Cheng, J. Bellaro, P. Benko, A. C. Snoeren, G. M. Voelker, and S. Savage, "Jigsaw: Solving the puzzle of enterprise 802.11 analysis," in *Proceedings of SIGCOMM 2006*, Pisa, Italy, Sept. 2006, pp. 39–50.

[18] Y.-C. Cheng, M. Afanasyev, P. Verkaik, P. Benko, J. Chiang, A. C. Soneren, S. Savage, and G. M. Voelker, "Automating cross-layer diagnosis of enterprise wireless networks," in *Proceedings of SIGCOMM 2007*, Kyoto, Japan, Aug. 2007.

[19] U. Deshpande, T. Henderson, and D. Kotz, "Channel sampling strategies for monitoring wireless networks," in *Proceedings of WiNMee 2006*, Boston, MA, Apr. 2006.

[20] J. Edney and W. A. Arbaugh, *Real 802.11 Security*. Reading, MA: Addison Wesley, July 2003.

[21] "Wireless vulnerabilities & exploits database," http://wirelessve.org.

[22] C. He and J. C. Mitchell, "Security analysis and improvements for IEEE 802.11i," in *Proceedings of NDSS 2005*, San Diego, CA, Feb. 2005.

[23] S. Radosavac, J. S. Baras, and O. Koutsopoulos, "A framework for MAC protocol misbehavior detection in wireless networks," in *Proceedings of WiSE 2005*, Cologne, Germany, Sept. 2005, pp. 33–42.

[24] "AirMagnet Enterprise," http://www.airmagnet.com/products/enterprise.htm.

[25] "Aruba Networks Air Monitors," http://www.arubanetworks.com/technology/air-monitors/.

[26] Y. Sheng, K. Tan, G. Chen, D. Kotz, and A. Campbell, "Detecting 802.11 MAC layer spoofing using received signal strength," Submitted to INFOCOM '08, July 2007.

[27] U. Deshpande, C. McDonald, and D. Kotz, "Coordinated sampling to improve the efficiency of wireless network monitoring," Submitted, May 2007.

[28] ——, "Refocusing in 802.11 wireless measurement," Submitted, May 2007.

[29] H. Yin, G. Chen, and J. Wang, "Detecting protected layer-3 rogue aps," in *Proceedings of IEEE BROADNETS 2007*, Raleigh, NC, Sept. 2007.