Proximity Breeds Danger: Emerging Threats in Metro-area Wireless Networks

P. Akritidis[*], W.Y. Chin[•], V.T. Lam[†], S. Sidiroglou[‡], K.G. Anagnostakis[•]

[•] *Systems and Security Department*
Institute for Infocomm Research (I^2R), Singapore
{kostas,wychin}@s3g.i2r.a-star.edu.sg

[‡] *Computer Science Department*
Columbia University, USA
stelios@cs.columbia.edu

[⋆] *Computer Laboratory*
Cambridge University, UK
Periklis.Akritidis@cl.cam.ac.uk

[†] *Dept. of Comp. Science and Engineering*
University of California San Diego, USA
vtlam@cs.ucsd.edu

Abstract

The growing popularity of wireless networks and mobile devices is starting to attract unwanted attention especially as potential targets for malicious activities reach critical mass. In this study, we try to quantify the threat from large-scale distributed attacks on wireless networks, and, more specifically, wifi networks in densely populated metropolitan areas. We focus on three likely attack scenarios: "wildfire" worms that can spread contagiously over and across wireless LANs, coordinated citywide phishing campaigns based on wireless spoofing, and rogue systems for compromising location privacy in a coordinated fashion. The first attack illustrates how dense wifi deployment may provide opportunities for attackers who want to quickly compromise large numbers of machines. The last two attacks illustrate how botnets can amplify wifi vulnerabilities, and how botnet power is amplified by wireless connectivity.

To quantify these threats, we rely on real-world data extracted from wifi maps of large metropolitan areas in the States and Singapore. Our results suggest that a carefully crafted wireless worm can infect up to 80% of all wifi connected hosts in some metropolitan areas within 20 minutes, and that an attacker can launch phishing attacks or build a tracking system to monitor the location of 10-50% of wireless users in these metropolitan areas with just 1,000 zombies under his control.

1 Introduction

The last two decades of network security research have demonstrated that attackers are continuously evolving, exploring creative ways to exploit systems, and targeting new technologies and services as they emerge. Indeed, the widespread use of email brought spam and email-viruses; broadband connectivity was followed by the rise of rapid self-propagating worms; while the growing use of online personal services and electronic commerce resulted in sophisticated personal data theft attacks, including phishing. Such trends suggest that any technology that reaches some kind of critical mass *will* attract the attention of attackers.

At the same time, modern attacks such as worms, spam, and phishing exploit gaps in traditional threat models that usually revolve around preventing unauthorized access and information disclosure. The new threat landscape requires security researchers to consider a wider range of attacks: opportunistic attacks in addition to targeted ones; attacks coming not just from malicious users, but also from subverted (yet otherwise benign) hosts; coordinated/distributed attacks in addition to isolated, single-source methods; and attacks blending flaws across layers, rather than exploiting a single vulnerability. Some of the largest security lapses in the last decade are due to designers ignoring the complexity of the threat landscape.

The increasing penetration of wireless networking, and more specifically wifi, may soon reach critical mass, making it necessary to examine whether the current state of wireless security is adequate for fending off likely attacks. This paper discusses three types of threats

[*] Part of this work was performed while P. Akritidis was visiting I^2R under an industrial attachment programme

[†] This work was performed while V.T. Lam was working at I^2R as a research engineer

[‡] Part of this work was performed while visiting I^2R

that seem insufficiently addressed by existing technology and deployment techniques. The first threat is *wildfire worms*, a class of worms that spreads contagiously between hosts on neighboring APs. We show that such worms can spread to a large fraction of hosts in a dense urban setting, and that the propagation speed can be such that most existing defenses cannot react in a timely fashion. Worse, such worms can penetrate through networks protected by WEP and other security mechanisms. The second threat we discuss is large-scale spoofing attacks that can be used for massive phishing and spam campaigns. We show how an attacker can easily use a botnet by acquiring access to wifi-capable zombie hosts, and can use these zombies to target not just the local wireless LAN, but *any* LAN within range, greatly increasing his reach across heterogeneous networks. Last but not least, we discuss the use of Tracknets, city-wide wifi botnets for unauthorized tracking of user location and behavior.

All three types of attacks, illustrated in Figure 1, are specific to wireless networks, and are based on the premise of dense wifi network deployment in urban settings. While most of the underlying vulnerabilities have been widely known for years, the amplifying power of densely deployed wifi networking has a profound impact on both the feasibility and the magnitude of the threats, suggesting that their importance may have been grossly underestimated. For instance, the susceptibility of open wireless LANs to spoofing has been well documented, but the need to be in physical proximity to the victim may have deterred the wider use of this attack so far. The ability to launch such attacks remotely is much more attractive, and can be scaled up by the use of coordination and a botnet infrastructure.

As a result of underestimating these threats, no countermeasures are currently implemented. The mechanisms needed to thwart these attacks are in most cases either available but not actively used, or not available but relatively easy to implement. The fact that such mechanisms are not used is of particular concern. For instance, 802.11i security mechanisms have been available for several years, and would address a large part of the problems described, but unfortunately they are currently not used by enough users. Similarly, the encryption of of MAC addresses would significantly increase the work-factor for Tracknets, but leaving the MAC addresses exposed was not deemed as a serious enough problem by the 802.11i group. Related to the worm problem, content-based filtering is widely used and intrusion prevention is a mature technology, yet to the best of our knowledge, it has not been employed in access point wireless-to-wireless forwarding Raising awareness on the threats, using convincing, experimental evidence, is therefore at least as important as exploring and implementing possible defenses.

The main focus of this paper is in quantifying these threats, specifically in metro-area wireless networks. We rely primarily on publicly available maps of wireless access point locations, also known as wardriving maps, and attempt to derive estimates on the feasibility and effectiveness of the attacks using measurements and simulations. These estimates paint a grim picture on the exposure of current wireless networks to such attacks, and indicate that the risks are further increased as wireless penetration continues to grow as predicted.

We also explore possible remediation strategies, most of which we have implemented and tested experimentally. In some cases, the defenses we have considered are just a matter of engineering, such as retrofitting reactive worm defense hooks and filtering capabilities in wifi gear. In other cases, countering the threat required novel techniques, such as those for detecting and preventing different variants of the basic spoofing attack – several such variants were discovered while pondering about possible defenses, and how attackers might try to circumvent them.

While some of these techniques would become redundant if 802.11i is widely deployed, we cannot rest on the assumption that such deployment will happen anytime soon, particularly in light of usability concerns. For example, none of the recently announced municipal wireless initiatives that we are aware of employ any form of protection, most likely due to the current perception of the risks of open wireless as well as the cost of managing accounts and passwords for large number of users – in one instance, 100,000 users and around 9M annual visitors. Furthermore, the choice of running an open wireless network may not always be a matter of ignorance or complacency, but a concious choice; for example, to provide network access to guests, backup connectivity to neighbors, etc [26]. Whether temporary or long-term, we believe that our supplementary defense techniques are useful for mitigating at least part of the threat.

2 Wildfire worms

The omnipresence and constantly improving capabilities of wireless mobile devices has attracted the regrettable attention of attackers, and in particular virus writers. The “Cabir” virus, which first appeared in 2004, was the first instance of mobile malware [27]. The virus exploited vulnerabilities in the Symbian OS and propagated through Bluetooth wireless connections. Experts predict the threat for smart phones and mobile devices is likely to increase significantly in the near future [40, 28].

Although such attacks *may* become prevalent in the years to come, in this paper we consider whether large-scale attacks are *already* feasible today on existing wireless infrastructure using current technology. In particular, we focus on worms that could spread entirely over

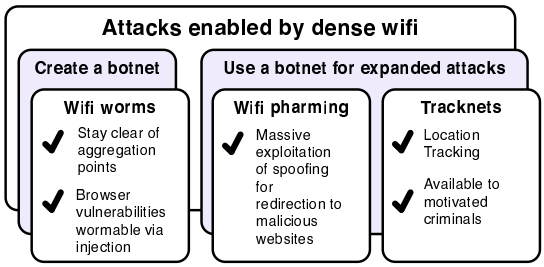

Figure 1: Dense wifi amplifies botnet threats.

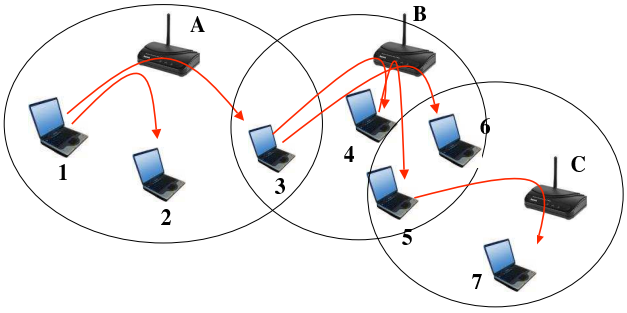

Figure 2: Simplified model of wildfire worm propagation.

802.11 wireless networks, even if such networks are completely heterogeneous. In this environment, the main concern is not necessarily the infection of mobile devices such as PDAs and cell phones, but the existing large population of laptops, desktops and other computers communicating over wifi. We consider worms that propagate entirely over wireless connections, trying to infect other computers tuned to the same access point (AP) and also other APs within range. A notable fraction of hosts in such an environment may also be mobile, and could therefore carry the infection from one AP to another. In densely populated metropolitan areas, it is conceivable that such a worm could infect a large fraction of wireless-connected hosts, especially considering pervasive vulnerabilities such as the ones exploited by Slammer [12], and recent browser vulnerabilities [13]. Such "client-side" vulnerabilities are of particular interest in a wifi setting, because unlike wired environments where a user needs to visit a malicious site to get exploited, it is often possible for an infected client to inject this kind of exploit via spoofing to any session between the target and a legitimate server. Considering the worst-case, a device driver exploit such as the recently discovered Intel driver attack [24, 36, 42] could carry the worm across platforms, and would even bypass VPN software which often blocks all local, wireless connections.

Although there has been considerable work in the literature on how to deal with large-scale attacks on traditional "wired" networks, there are at least three differences between wireless networks that require alternative solutions. First, wireless attacks can spread contagiously over wireless links based on proximity – similarly to real-world diseases – in contrast to the any-to-any communication possible over the Internet. This renders previous models and analyses of Internet-based worm propagation ineffectual as they cannot be directly mapped to wireless networks. Second, traffic in wireless networks is difficult to control using conventional methods, in lack of "hard" enforcement points such as firewalls between the communicating nodes. This is likely to significantly constrain the space for potential defenses. For instance, if such a wireless worm were to be unleashed today, it would most likely go undetected by most, if not all, current attack detection infrastructures [17, 2, 3]. Finally, devices (*e.g.* handheld devices in the near future) in these environments are likely to be significantly more resource-constrained, at least in contrast to traditional desktop settings, and it is therefore more difficult and expensive to employ end-point security measures.

This paper is not the first to examine the threat of worms in wireless networks. Other researchers have made attempts at deriving contagion models in MANETs, examining viruses that spread according to user mobility, or measuring propagation dynamics in a campus network (these studies are discussed further in Section 6). Our paper is first to explore, in depth, the problem of wildfire worms and proximity propagation in densely populated areas. Specifically, we discuss the threat of worms that propagate entirely over wifi connections, and attempt to quantify the threat in terms of infection prevalence and infection timescales. Providing reliable estimates of potential infection prevalence is important for creating awareness on the severity of the threat, while the likely infection times are needed to guide the design of suitable countermeasures. Our analysis relies on simulated outbreaks of wifi worms driven by real-world data derived from wifi maps of large metropolitan areas around the world. Among other observations, our results suggest that a carefully crafted wildfire worm can infect all vulnerable wifi-connected computers in 80% of access points in some studied areas within 10-20 minutes – timescales at which traditional defenses may not be able to react in a timely fashion.

In this section, we describe the design and attack vectors of a wifi worm. The fundamental principle is that a wildfire worm relies on local, proximity-based propagation within shared medium broadcast environment such as WLAN.

2.1 Wifi worm propagation

Figure 2 illustrates the propagation dynamics of wildfire worms. Three access points A, B and C provide wire-

less coverage to end users, e.g. mobile nodes 1–7. They could represent, for example, WLANs deployed at adjacent buildings. Note that overlapping usually exists between adjacent access points for both residential networks (especially in densely populated cities) and corporate wireless networks (to allow for continuous connectivity and seamless mobility roaming).

Assume node 1 is the initial source of infection, *i.e.* it was infected previously at some other location before associating with access point A. Once activated, the worm analyzes WLAN A and probes all victims in the neighborhood; hence node 2 and node 3 eventually get infected. Note that node 3 is under coverage of both A and B. Normally node 3 picks and associates with only one access point, which is decided by certain criteria such as wifi signal-to-noise ratio. A worm-infected node, however, can gather a list of usable access points within reach and scan them for victims in the proximity. Effectively, the worm toggles association between usable WLANs to spread itself. Eventually all nodes in WLAN B and C are compromised through node 3 and nodes 5/6 respectively.

Nodes at coverage intersection of access points are "bridges" that help propagate the worm. These nodes can be thought of as "connectors" in the small-world phenomenon hypothesis [44, 41]. Contrary to the context of traditional Internet worms in which node 1 could probe and infect node 7 instantly, propagation dynamics of wildfire worms are similar to gradual and local diffuseness of disease. Therefore, a major advantage and difference of a wildfire worm over a regular Internet worm is that a wildfire worm can propagate entirely locally within each connectivity area, and thus evade firewalls and intrusion detection/prevention systems located at traditional enforcement points on the boundary between the local networks and the Internet.

Fertile ground for wildfire worms are wireless hotspot networks, which provide Internet access in public areas such as restaurants and airports, and private wireless networks of home users in residential areas. For example, Singapore government is realizing a "Digital Singapore" with wireless hotspots available at every street corner where people can log onto the Internet and receive emails on the move. Section 2.6.2 evaluates whether wifi penetration in metropolitan areas is sufficient for sustaining the spread of a wifi worm.

2.2 Mobility

Presently, the wireless node population consists mostly of laptops, and to a lesser extent of PDAs and smartphones (including wifi VoIP phones). The mobility patterns of wireless users can affect worm dynamics in three ways. First, mobility could compensate for sparse connectivity that may hinder wildfire-style propagation, as users carry the worm to networks previously unreachable by the worm. This is not restricted to just the places where the user turns on the laptop, as Laptops can also be programmed to wake up periodically as the user moves from one place to another At the same time, user mobility also helps worm propagation into protected networks, whether they use WEP or more secure WPA/WPA2 protection, as the user will voluntarily (and perhaps even automatically) authenticate to those networks. Finally, the worm could create fake access points to lure and infect mobile users.

2.3 Open vs. Protected Access Points

There is a significant number of publicly available "open" access points; the rest are protected with Wired Equivalent Privacy (WEP) encryption or Wifi Protected Access (WPA). A worm can propagate over unprotected wireless networks in the way shown in Figure 2. Moreover, as a result of design and implementation flaws, WEP encryption is insecure. There is a handful of WEP attacks in the literature, e.g. weak IV attacks [30], keystream re-use [15, 22] and more recently fragmentation attacks [20] . These attacks are not just of theoretical value; they have been implemented into many practical and efficient WEP cracking tools freely available on the Internet. Wepcrack [8] did a performance comparison on some of such tools. Among them, Aircrack [1] is particularly powerful with a high success rate and relatively low cracking time that could vary between 5 seconds to 1 minute. However Aircrack needs to spend considerable time to sniff and capture sufficient wireless packets before cracking attempt. For example, after analyzing wireless usage statistics at a university campus [7], we determine that it may take 1-2 hours on average to successfully crack WEP encryption. Instead of passively sniffing packets, the worm could also employ active attacks e.g., discovering the encrypted version of a plaintext packet [8]. As for WPA, while not inherently weak, it is susceptible to bruteforce attacks if used with a weak password in the most common WPA/PSK configuration. Given the apparent susceptibility of the currently available protection mechanisms, it seems likely that worms would consider carrying the additional payload of including cracking tools.

2.4 Infection process

In the design of a wildfire worm, we note that there are two possible ways to exploit vulnerabilities. The first approach, known as the "push method", is to directly probe for an exploitable service and inject code to that service on clients just as traditional worms (e.g. DCOM RPC vulnerability on port 135 for Blaster worm). With the second approach, dubbed "pull method", instead of relying on a service vulnerability, the attacker exploits vulnerabilities, such as browser vulnerabilities by per-

forming a man-in-the-middle attack. For example, the infected node can listen on the wifi and wait for the victim to make a DNS request, spoof the response pointing to itself (or some other, unused address), pretend it is the web-server and respond with pages that include exploits such as the WMF exploit [13] or other exploits for IE and Mozilla that attempt to execute malicious code. ARP spoofing and TCP injection attacks may be used as well. We note that the distinction between worm and virus is blurred in this case, as propagation may require some form of user interaction, yet the attack is piggybacked on communication to a third party, rather than between infected and targeted host. The broadcast nature of most wireless setups makes "pull" attacks attractive for wildfire worms as they can be exploited at a scale that was never possible for Internet worms.

2.5 Proof-of-concept implementation

We have implemented a proof-of-concept wildfire worm for both Windows XP and Windows Vista. This worm, dubbed Wildfire/A, has been submitted to security vendors for testing. The implementation of this worm was surprisingly straight-forward given the plethora of tools publicly available.

The WLAN API available for both Windows-Vista and -XP facilitates the process of managing AP association and scanning. Through this API, the worm is able to actively scan for open "visible" APs and, in turn, associate with them. Once associated with an AP, the worm scans the local subnet for vulnerable machines. For this particular proof-of-concept implementation we only considered push exploits, namely, the chunked-encoding vulnerability found in the Apache Web server 1.22. The worm payload is packaged as a self-extracting archive that contains the libraries required by the WLAN API as well as a copy of the actual worm. We have confirmed that the worm operates as expected in a small scale experiment with 4 APs and 15 vulnerable hosts.

2.6 Analysis

As with all worms, wildfire worms need to exploit a vulnerability to infect end-hosts. Unlike Internet worms that can effectively spread even if the vulnerable population is very small [48], wifi worms depend on the vulnerability being widespread. This raises two questions: what critical mass does a wildfire worm require to be effective, and whether there are indeed such pervasive vulnerabilities.

2.6.1 Vulnerabilities

To determine whether there is a significant number of pervasive vulnerabilities, we analyze vulnerability data from a variety of sources, including NVD [4], Securityfocus [6], and other independent sources. We focus on remotely exploitable vulnerabilities in the default installation of Windows XP Service Pack 2, between August 2004 (the Windows XP SP2 release date) and January 2007. We classify vulnerabilities based on whether they can be triggered through direct injection ("push" exploits) or through spoofing attacks as discussed in the previous section ("pull" exploits). Starting from basic information available through the NVD database, we verify the vulnerability information and derive further details such as exploit availability, exploitation technique, disclosure date, and patch dates primarily from Securityfocus archives but also other independent sources.

For all the qualifying vulnerabilities, we attempt to get a rough estimate of the *vulnerability window*: the amount of time the vulnerability was known and not patched in the majority of hosts. Unfortunately, publicly-available information does not always give us an accurate timeline of exploitation time vs. disclosure time, and we therefore have to make certain assumptions. In particular, we optimistically assume that by the time a vendor (in this case, Microsoft) releases an update, all hosts in the network are instantly updated and patched. In most (but not all) cases, the vulnerability is disclosed by the vendor only when the update is available. As such, it is not always possible to determine exactly when the vulnerability became known and to consider this as the start of the vulnerability window.

In lack of more accurate data, we assume that the vulnerability window starts two week before the update is issued, as Microsoft only posts updates every second Tuesday of each month. This is corroborated by Symantec which reported an average period of 13 days for the first half of 2006 between disclosure date of a vulnerability and the release date of an associated patch by Microsoft [53].

The results indicate significant exposure to vulnerabilities in the default configuration over the last two years, accounting for more than 50% of all days in the total period. Vulnerabilities of "push" type, i.e., that affect services and don't need user interaction, were active for 105 days (11.89%) while "pull" type, i.e., that need user-interaction of some-kind, were active for 428 days (48.47%). We believe this observation suggests a trend, in which server/services components seem to be relatively robust when compared to client components. This is especially alarming in the context of wifi worms, because they are particularly suited for exploiting such vulnerabilities, and their abundance may give them another evolutionary advantage over Internet worms. Overall, we have found that 60% of the listed vulnerabilities had public exploits available for 391 days (44.28%) during the time period.

Other analyses of vulnerability exposure for the years 2004–2006 published on the Internet paint an even dimmer picture for "pull" type attacks. For a total of 284

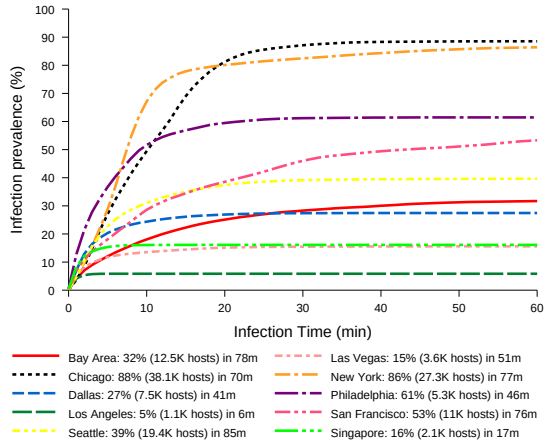

Figure 3: Spread of a wild-fire worm.

days (78%) in 2006, exploit code for known, unpatched critical flaws in pre-IE7 versions of the browser was publicly available on the Internet, and there were at least 98 days in which no software fixes from Microsoft were available to fix IE flaws that criminals were actively using to steal personal and financial data from users [39]. For at least 256 days (70%) in 2005, Internet Explorer contained unpatched vulnerabilities where the exploit method had been publicly disclosed but was not necessarily being used, and for at least 38 days in 2005, IE was vulnerable to unpatched critical security flaws that were being actively exploited [38]. A fully patched Internet Explorer installation was known to be unsafe for 98% of 2004, and for 200 days (54%) there was a worm or virus in the wild exploiting one of those unpatched vulnerabilities [11]. For Firefox, there were 56 days (15%) in 2004 where a publicly known remote-code execution had not yet been thwarted with a patch [11].

2.6.2 Worm simulation

To understand wildfire worm propagation, we simulate the outbreak of a worm in nine well-known US cities and Singapore. For this we relied on publicly available maps of Access Point locations from the *Wigle.net* [10] "wardriving" database, as well as empirically derived data for the city of Singapore. From these maps, we only consider open APs where the worm can spread without having to crack the encryption or the password.

Available war-driving maps chart APs but not connected hosts, so we had to populate them by randomly distributing hosts around APs. Based on our war-driving measurements and assuming a pervasive vulnerability, we distribute an average of 0.5 hosts per AP with Poisson distribution at an exponentially distributed distance of 10 m on average. We model effective AP range as omnidirectional with a radius of 90 m.

Finally, we do not consider the possibility of bypassing the AP to directly infect hosts within range using low level techniques because these depend on the available device driver and may not be widely available. We also ignore host mobility except that we assume the epidemic starts from 50 random locations to avoid artificially confining the worm to a sparse disconnected portion of the city.

The infection time for one hop is determined by four factors: scanning time, association time, IP acquisition time and transmission time. Based on our wildfire worm prototype, we assume a scanning and association time of about 1.5 seconds. We do not model DHCP interaction in our simulations as the worm can simply hijack an IP address. With an effective throughput of 14 mbps and 8 mbps for typical 802.11g and 802.11b networks respectively, the transmission speed is between 1 Mbytes and 1.7 Mbytes per second. Since the bandwidth will be shared among hosts, each host gets a transmission speed of a few hundreds kbytes/seconds. We assume a transmission speed of 100 kbytes/sec per host. For a worm size of 100K – which should be sufficient – the transmission time is about 1 second. A simulated worm-infected node infects its neighbours sequentially using these parameters.

Each simulation consists of 20 runs; for each run we start the infection from 50 different randomly selected hosts. We collect the mean values across runs of infection prevalence over time.

Figure 3 is a plot of infection prevalence over time for a "push" worm. Dense cities are infected very fast: 80% of New York and Chicago in less than 20 minutes. San Francisco and Philadelphia are infected fast as well: about 50% of San Francisco and Philadelphia are infected in 45 and 11 minutes respectively. A wildfire worm does not spread significantly in Los Angeles and Las Vegas, but on a longer time scale a worm could still spread with the help of user mobility. The worm can spread fast as long as there are enough APs to maintain connectivity, but high density may even bog down the worm in some cases. In absolute numbers, we see that an attacker could quickly gain access to ten's of thousands of hosts in most cities. The attacker could start simultaneous, independent epidemics in many cities using the Internet to infect a few seed hosts.

As for pull worms, we briefly summarize the simulation results here without a figure. Their simulated spread is limited compared to push worms – prevalence of pull attacks is limited to 60% in 3 hours for New York and Chicago, but they are potentially more dangerous, as they can take advantage of more vulnerabilities. They are slower because the infection time must include waiting for the victim to offer an opportunity for infection in the form of a DNS requests or TCP connection. On the

other hand, the worm can wait in parallel for any victim to become active. We use a very rough estimate of 10 minutes for waiting time to get an idea of the time scales involved, acknowledging that some machines may have no browsing activity at the time. The pull worm also requires higher density since we assume a shorter range of 60 m. Weaker antennas and increased interference typically weaken client transmission characteristics when compared with APs.

Overall, these time-scales suggest that automated defenses are crucial for defending against wildfire worms.

3 Large-scale Wifi Spoofing

One key property of open 802.11 networks is that they are built around a broadcast medium, where any wireless station can transmit wireless frames, and can listen to all other frames transmitted on the network. This is reminiscent of shared Ethernet segments of the 90's.

This property makes wireless LANs susceptible to spoofing and injection attacks, as discussed extensively in the context of wired Ethernet (but effectively disappeared with the emergence of switched Ethernet). The basic idea is that an attacker can monitor the communication between hosts on the wireless network, or between a host on the wireless network and an external party. If the communication is not properly encrypted, the attacker can elicit session state through eavesdropping, and if the communication is not authenticated, he can then *inject* frames to one session endpoint pretending to come from the other session endpoint.

Most protocols, such as DNS, DHCP and TCP are susceptible to this attack. In the case of DNS, the attacker can watch for outgoing DNS queries and inject responses pointing to a host under his control. For TCP the attack is similar – all the attacker needs to know is the current state of the connection in terms of sequence numbers. At connection setup, he may even completely take over the connection by injecting the proper SYN-ACK, resulting in the legitimate endpoint being out of sync. Injection is also possible at any point in the connection as long as the attacker can time injection attempts to properly deliver TCP segments to the victim network stack. The DHCP protocol can be spoofed to have a victim use an IP address and default gateway that gives the attacker full control over all of his traffic. However, it may be less attractive than DNS and TCP spoofing as the attacker has to wait for the victim to refresh his DHCP lease, or else attack only hosts that have connected after the attacker has obtained access to the wifi network.

While in the 90's such attacks were seen as enablers for unauthorized access, in today's threat landscape they are more likely to be used for "modern" attacks such as phishing, spam and exploit injection. In the previous section we briefly discussed how injection can be used to propagate a worm through client-side vulnerabilities. In this section we focus on spoofing primarily for the case of launching phishing attacks, and discuss ways to detect and prevent them. DNS spoofing is highly attractive for phishing as, for example, the attacker may set up a mock banking website that would relay manipulated requests to the real site in a man-in-the-middle fashion. We note that in this case, two-factor authentication cannot help. Similarly, TCP injection can be used to insert redirection instructions, advertisements, or spam to otherwise legitimate Web pages. Sophisticated attacks can even subvert user's services, such as using a victim gmail account, etc.

The use of such techniques in wifi for phishing has been documented previously. The so-called "parking lot attack" involves the attacker being in physical proximity to the target network. While this attack may be interesting by itself, we are not aware of any extensive use of this technique. One main disadvantage is that the physical proximity constraint increases the risk to the attacker, especially in environments with pervasive CCTV coverage that can be used for forensics. In the context of this paper we explore how proximity enables remotely controlled bots to be used for such activities. In this case, the attacker can acquire access to a wifi-enabled host located in a wifi-rich location. In contrast to traditional Trojans, the attacker need not try to elicit information from the owner of the actual machine that is being exploited. Rather, the attacker may perform spoofing on *any* wireless network within range from the host under his control using channel hopping and/or temporary association for the duration of the attack. The dense use of wifi in metropolitan areas makes this model quite attractive, as it may significantly *amplify* the attacker's capabilities.

3.1 Analysis

To determine the effectiveness of spoofing attacks in terms of scale we rely on the same publicly-available wifi maps used for analyzing wildfire worms. We attempt to get a rough estimate of the number of access points that hosts on the map can connect to. As we only have access point locations, we add hypothetical hosts within range from each access point. We distribute 1 host per AP and assume a communication range of 60m.

We compute the number of neighboring APs for each host, that is, all APs within range excluding the AP it is directly connected to. We consider only "open" APs that do not use any wireless security protocol, even though the attacker may well be able to crack into WEP-enabled networks using well-known attacks and tools.

The results for our analysis on 10 different metro areas are shown in Figure 4. We see that in half of the cities 90-99% of all hosts can connect to at least one more neighboring AP; 20-50% of hosts can connect to at least 10

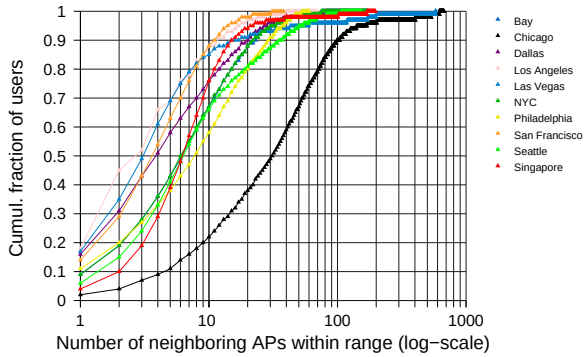

Figure 4: Number of WLAN networks observable from random hosts in metro areas (range 60 m).

attacks \ defenses	Ingress Filtering	Packet Rewriting	802.11 Spoofing Detection	Inconsist. Duplicate Detection	Whisper Attack Detection
Baseline Spoofing	✓✓✓		✓✓	✓✓	✓
External Collaborator		✓✓✓		✓✓	✓
802.11-level Spoofing			✓✓	✓✓	
Whisper Attack					✓

Figure 5: Spoofing defense space.

additional APs; and a small but non-negligible number of hosts, as high as 10% in Chicago are within range of more than 100 APs. Unsurprisingly, the results are worse for Chicago, which seems very densely populated, and less so for relatively sparse areas.

Overall, the results confirm our fear that controlling wifi-enabled hosts in densely populated areas can be highly attractive to attackers.

4 Wifi tracknets

The proliferation of city-wide wifi networks has already raised serious concern over privacy implications. Privacy advocates fear that wifi networks can be used to record location information for the operating ISPs, their partners, and possibly law enforcement, raising concerns that wifi can be used to track general user behavior in a ”Big Brother” fashion.

However worrying this scenario might appear, it can be classified as a mere nuisance when compared with the possibility of *anyone* being able to remotely set up a tracking system, without even having to set up physical infrastructure. Such systems, which could be termed as *Tracknets* can be deployed using a reasonably sized botnet, providing a user-tracking mechanism that can operate across wireless network boundaries. Criminal gangs are known to operate marketplaces for bots, sometimes with specific features such as high bandwidth and CPU power, priced between $1 and $40 per compromised PC according to security exprerts who have monitored IRC chat room echanges [54]. It is conceivable that attributes such as wifi connectivity, and location within a metro-area could be added to the list of features to facilitate attacks such as those described here.

Such a botnet can then track location information [16], possibly coupled with user-profiles that can span across heterogeneous wireless LANs. The location of the zombies comprising the bot can be infered from the ESSID of their AP using public wifi maps. (In fact, this service is already provided by companies such as Navizon and Skyhook.) The number of users that can be tracked using Tracknets and its coverage are commensurate with the size of the botnet population and the amplifying effect of proximity, similar to the spoofing threat discussed in the previous section.

Several services can leak significant amounts of privacy-sensitive information. This information can, in turn, be used for targeted Phishing and spam attacks, blackmail, and for pre-attack reconnaissance such as building hit-lists. In addition to high-information-leak vectors, several techniques can provide personal information at a lower granularity that might not be able to distinctly identify individual users but can be used to classify sets of users according to broader set of criteria such as OS version version, wireless driver information and general browsing behaviour. In this section we briefly examine some of the most obvious tracking vectors. Our investigation is far from exhaustive and only scratches the surface of possible ways that users could be tagged and tracked. Nevertheless, the vectors we discuss show *at least* one set of techniques that seem threatening enough by themselves, and may be representative of other approaches.

MAC address The obvious way to track users across heterogeneous WLANs is to use the MAC address as unique identifier. Trackers can use this information to correlate any other behavioral information to a MAC address to easily create profiles. Fortunately, although MAC addresses are permanent by design, there exist a number of mechanisms that allow users to change the identifier. Gruteser et al [32] introduce the idea of short-lived disposable MAC addresses as a technique for the reduction of the effectiveness of location tracking. However, randomizing MAC addresses often leads to problems. For example, several ISPs use MAC addresses

to map IP addresses. Also, some software licenses are bound to a specific MAC address. Furthermore, even in the presence of such techniques, user profiling can still effectively track users in dense urban environments. In our system, we use MAC addresses as temporary identifiers for correlating information that will be used to create user profiles as described below.

Live bookmarks – RSS Live bookmarking is a new popular method for displaying web feeds as bookmarks. Its popularity surged when it was introduced in Mozilla Firefox 1.0 back in 2004 and can now be found in several other popular web browsers such as Apple's Safari and Internet Explorer 7. Live bookmarks subscribe to user-defined RSS feeds and are periodically updated so as to display the latest articles. The ability to customize feeds along with the inherent periodicity of the updates make Live Bookmarks susceptible to eavesdropper profiling. In particular, as users subscribe to more RSS feeds they inadvertently create distinct profiles that can be used to track them. Given the wide range of tools available for parsing RSS feeds, it is trivial for a tracker to parse the feeds so as to extract user personalization in addition to RSS subscription information. Worse, by using traffic analysis to identify such communications based on their periodicity and creating a signature based on packet size distributions, an attacker could possibly track users over encrypted WLANs, however, we have not investigated this scenario further.

Tracknet bots would collect and parse all requests to RSS feeds. The information derived from the feed is then associated to an individual node. The node is temporarily identified by IP and MAC address for the current session. Any other information that is collected from the particular node is collected in a tracking tuple that correlates all other pertinent fields that aid in the identification of the node. In order to reduce the number of identification false positives we correlate the RSS fingerprint with the base station ESSID. Distinct fingerprints that appear at the same location (*e.g.* home or workplace) might point to a distinct identify with a higher level of confidence.

Location tracking Collaborating bots can use radio signal characteristics of WLANs to determine a user's location with relative accuracy using triangulation techniques. This information, in combination with other extracted personal information can lead to considerable privacy leaks. Specifically, bots can use this information to infer user behavior. For example, information on entertainment habits, political orientation, medical information can be potentially derived.

Other services Beyond the mechanisms described above, there are numerous other protocols and services that leak significant personal information. For example, numerous Instant Messaging (IM) system do not employ encryption so all user identification information is available to eavesdroppers. Although this information might not be significant on its own, when it is correlated with other sensitive information, it can be used to construct a distinct user profile. Other systems that can be used to fingerprint user behavior are the mail servers that users connect to, information from other networking protocols such as NETBIOS and AppleTalk and even which VPN servers a user connects to.

The growing popularity of Google and other online service portals, has moved a number of user services to central aggregated locations where users can check their RSS feeds and email. Although this configuration changes the network fingerprint that is emitted by services it does not reduce the amount of information that is leaked. For example, the Google homepage includes links to personalized RSS feeds including the user's email address in plain text, which often points to a user's real identity, e.g., `john.doe@gmail.com`. This information can be readily used to create very accurate user profiles since a tracker can intercept these unencrypted HTTP transfers.

Another serious vector of information leak is (to no surprise) the use of cookies. Cookies are used extensively as a mechanism for servers to identify users and track their access. The threat of Cookies to user privacy has received considerable attention in the literature [23]. In the context of tracknets, the exchange of Cookie information can be used to extract personalized user information based on both the contents of the Cookies and their transmission fingerprint. For example, Google, a company synonymous with Internet search uses cookies that expire in 2036. The cookie uses a 16-digit identifier to track user preferences and, inevitably, track user behavior. Given the popularity of the search engine, it is not unreasonable to assume that a large percentage of the user population will emit this identifier during its lifetime, adding another mechanism for user tracking.

The Dynamic Host Configuration Protocol (DHCP) is a ubiquitous protocol used for automating network configuration. Unfortunately, there is no privacy protection for DHCP messages, so an eavesdropper who can monitor the link between the DHCP server and requesting client can discover the information contained in this option. For example, the following snippet illustrates the kind of information that can be derived from a DHCP request. Information on the types of services and more importantly hostname information is made readily available to eavesdroppers.

```
Client IP: 10.50.16.205
Client Ethernet Address: 00:17:f2:40:61:65
Vendor-rfc1048:
DHCP:REQUEST
PR:SM+DG+NS+DN+NI+NITAG+SLP-DA+SLP-SCOPE+LDAP+T252
MSZ:1500
```

```
CID:[ether]00:17:f2:40:61:65
LT:7776000
HN:"alamak"
```

We collect and correlate the information derived from DHCP headers. In particular, we are interested in user-identifying information such as the user's hostname. This information might appear innocuous but is often linked to personal information such as the user's name or company information. Again, in this case we associate DHCP-derived information with the base station's ESSID.

4.1 Experimental analysis

We determine how effective an attacker can be in tracking users using a botnet consisting of wifi-enabled hosts within a metropolitan area. For this purpose, we rely on the same wifi maps used for analyzing the worm and spoofing attacks. The effectiveness of a tracknet can be expressed in terms of *coverage*, that is, the fraction of wireless LANs that are within range from a given set of subverted nodes participating in the tracknet. The feasibility of a tracknet also relates to the number of subverted nodes that the attacker needs to obtain in order to achieve a certain level of coverage. As the attacker may have little control over which hosts to subvert (or buy access to) and where they are located, in each experiment we assume a random subset of hosts on the wifi map. As MAC addresses are exposed even when the network uses WEP or 802.11i encryption, we consider all access points regardless of whether they are open or protected – in other words, a tracker can monitor *any* network within range.

The results for 10 metro areas are shown in Figure 6. We observe that the fraction of subverted hosts needed to track users is relatively modest: with hosts on just 1% of all APs in a dense area, a tracknet can cover between 5% and 40% of all traffic. As expected, full coverage is not easy to achieve, but having trackers on around 7% can reach between 30% and 80% coverage. As with the worm and spoofing threats, the high density of Chicago and NYC make them particularly susceptible to this attack: less than 1,000 zombies are sufficient to cover 40% of the APs.

At the time of writing this paper, all MAC addresses are exposed, but it is worth investigating whether using disposable MAC addresses would help address this problem. As discussed previously, we are particularly concerned about other high-information-leak profiling techniques that could essentially offer uniquely identifying information equivalent to a MAC address. We focus on RSS feeds as one emerging source of leaks, and try to quantify the ability of an attacker to use this information for tracking purposes. For this purpose, we have obtained from an online service provider the set of RSS feeds that users are subscribed to, for around 100,000 users. The size of the dataset is important as we seek to measure the *uniqueness* of each RSS profile. We therefore measure for each user, whether any other users have the same exact profile, in which case we say that we have a profile collision (which could make tracking information ambiguous and confusing to the attacker). As some users have empty or very small profiles, we expect more collisions there, and we therefore compute collision statistics for those users with at least a minimum number of feeds in their RSS set.

The results are presented in Figure 7. As expected for a minimum RSS set of zero, that is, no constraints, the fraction of users with colliding profiles is around 30% – most of them are users with an empty profile. Removing only those that have an empty profile, that is, focusing on a minimum set of one entry, the collision probability is 0.02 to 0.07, significantly lower and reasonable enough to allow a tracknet to identify a user with high confidence, especially given that this information can be correlated with other data. For users with more substantial RSS feeds, the collision probability is between 0.002 and 0.01, indicating highly unique profiles. The scaling behavior of collision statistics is of particular importance here: we see that collision probability increases with the number of RSS profiles in the dataset, yet the difference seems to be small between a database of 50K users and a database of 100K users. If a tracknet is supposed to cover a whole city, the number of profiles can be much larger than the set we considered here, but our results suggest that collision probability is unlikely to worsen significantly. Furthermore, when a user's RSS fingerprint is coupled with location information such as mobility patterns, this set can be reduced even further.

5 Defense strategy

The threat of wildfire worms and large-scale spoofing can be reduced significantly with the use of existing wireless security standards such as WPA/WPA2, with strong encryption and hard-to-guess passwords. Unfortunately, despite the wide availability of such techniques, users do not seem to employ them. Even if this is simply because there have been no large-scale attacks yet, the use of passwords hinders usability and robustness. It is likely that even if such measures are implemented, in many cases the passwords are not going to be strong enough to resist brute force attacks. As such, it seems worthwhile investigating alternative, reactive defenses specific to the attack vectors discussed so far. In the remainder of this section we discuss such defenses, as implemented in a prototype system for automated defense against wildfire worms and spoofing attacks based on the Linksys OpenWRT [5] router and optionally using an external controller and centralized threat analysis.

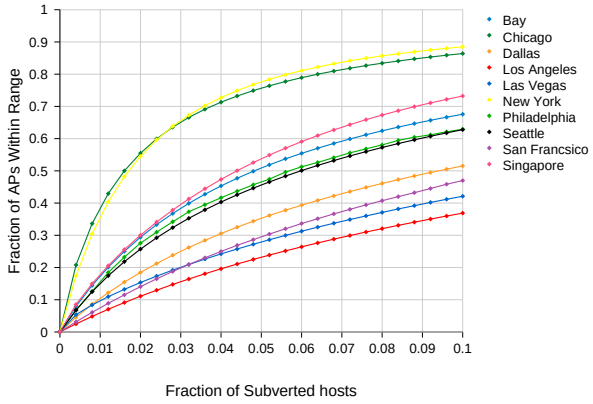

Figure 6: AP coverage of a given fraction of random bots in a metro area assuming a range of 60 m.

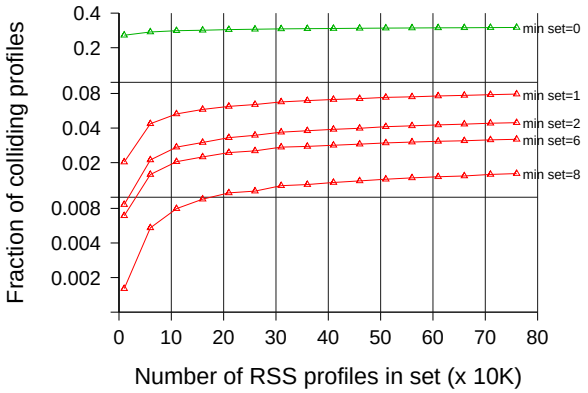

Figure 7: RSS set uniqueness.

5.1 Wireless IPS

In our implementation, we have adapted the snort IPS [46] to run on OpenWRT. While previous implementations have used snort to filter traffic between the wireless network and the external ethernet connection, our implementation disables the normal low-level wireless-to-wireless forwarding and uses ebtables and IPtables to redirect traffic through userland where it can be processed by snort.

As APs typically have limited computing resources, it may not be possible to have a fully fledged IPS running on them. Increasing their capabilities may also be prohibitively expensive. There are at least two possible options to address this problem. The first option is to use only a subset of the signatures, most likely signatures for attacks against vulnerabilities that may not be universally patched yet. The second option is to implement the IPS functionality in a centralized wireless controller, and have the APs forward all local traffic for inspection before retransmitting to the wireless medium.

The main advantage of using a wireless controller is that it provides flexibility for devoting more resources to traffic inspection. It is also consistent with industry trends towards cheap, "dumb" access points managed by a wireless switch. However, none of the wireless switches we are aware of provide any filtering capabilities for internal WLAN traffic such as wildfire worms. In our case, the additional wireless-to-wireless IPS functionality is implemented as a standalone wireless controller. This functionality can be retrofitted into wireless controllers or implemented as part of a secondary controller – protocols for AP to controller communication are being standardized, and thus interoperability is likely to be achievable.

For zero-day attacks for which there are no signatures, we rely on honeypot feeds from access points back to an analysis center. There, we use the Argos system [45], which uses dynamic taint analysis to trap the execution of remotely-injected code, for detection coupled with our custom signature generation system. Our system collects packet trace samples corresponding to the exploitation attempts detected by Argos and then uses a heuristic for generating network-level attack signatures in the form of simple patterns. The heuristic tries to identify a substring that is sufficiently large, sufficiently frequent in the attack samples and sufficiently infrequent in benign traffic. This last part is important for addressing concerns about false positives as well as attempts to manipulate signature generation for denial-of-service purposes. Our implementation uses a novel inverse indexing scheme on previously collected packet traces. While in our test setup these traces are maintained centrally at the threat center, it is conceivable that such testing can be performed at each site independently. These signatures are then installed on the AP or the wireless controller as snort sensor rules.

Filtering of internal WLAN traffic assumes that the worm does not tamper with the wireless device driver and firmware. If such tampering is possible, the attacker may spoof access point transmissions directly – for which tools are publicly available [9], and bypass filtering mechanisms applied to traffic relayed over the AP. The AP can detect attempts to impersonate it as long as it can pick up the messages sent by the attacker, but this leads immediately to another attack scenario: the attacker can hide his emissions from the AP by tuning the wifi radio power or using directional antennas so that the spoofed packets can reach the victim, but cannot reach the AP (or external detection device). We refer to this as the *whisper* attack. The attack seems difficult to engineer, as it requires both the low-level driver/firmware

hacks of the basic 802.11 spoofing attack, as well as careful tuning of the radio. Unfortunately, newer chipsets provide improvements in power control, and it is likely that the attacker can easily find the "right" power setting to launch the attack by probing both the victim and the AP with different power settings, all controlled through the driver API. In some cases, the relative positions of AP, victim, and attacker may prevent this attack. In addition, not using an AP to relay frames limits the communication range. Using power control to evade the AP may limit the range even further, to the point where it may become impractical to perform whisper attacks in the context of the massive attacks we discussed.

5.2 Spoofing defense strategy and attack-defense co-evolution

Assuming WPA and VPN solutions comes with a considerable usability cost; we investigate lightweight alternatives. Interestingly, our exploration of defenses against spoofing attacks has revealed a small arms race. In particular, while developing defense techniques we discovered several new variations of the attack, each defeating one of our countermeasures. In this section, we discuss the attacks, the countermeasures and present results evaluating their effectiveness. These findings are summarized in Figure 5. We focus on DNS spoofing for simplicity, but in most cases the attacks and countermeasures are similar for other protocols.

5.2.1 Wireless ingress filtering defense

As discussed previously, the simplest form of DNS spoofing involves the attacker lurking for DNS requests to the target site, and then injecting a fake DNS response pointing to a site under the attacker's control. It seems straightforward to defend against this attack through the use of *ingress filtering* at the AP. Ingress filtering ensures that all traffic broadcast by the AP on the wireless network is checked in terms of IP address and the interface on which it is received. That is, traffic originating from the wireless network should have IP addresses on the local wireless network. (Similarly, but less relevant here, traffic from the external network should not have an IP address on the internal network.) A DNS request is usually sent to a resolver outside the wireless LAN, and therefore the DNS response is expected from an external address. A spoofed response is trivial to detect, as it arrives on the AP from the wireless interface and has an external IP address.

5.2.2 External collaborator attack

A variation of the spoofing attack that circumvents ingress filtering involves the use of an external collaborator. In this variation of the attack, the attacker is again eavesdropping on the wireless LAN lurking for DNS requests, but instead of sending the spoofed response from the wireless LAN, signals another host on the Internet to send a spoofed response to the victim. Being able to eavesdrop is crucial, as it allows the attacker to relay the needed DNS identifier and port number information to the remote collaborator.

There are two constraints for the attacker that make this attack more difficult. First, the remote collaborator needs to be able to send packets with the source IP spoofed. Unfortunately, a recent study [18] shows that spoofing is still possible on more than 30% of hosts due to the limited use of source filtering. Second, the remote collaborator needs to send the spoofed DNS response before the legitimate DNS response arrives. Thus, the attacker would need to locate a collaborator that is closer by in terms of round-trip times.

5.2.3 Packet rewriting defense

One way to defend against this attack is to rewrite packets as they flow through the AP to the outside world, mapping the DNS id and port number, TCP sequence numbers, etc., to a different space, then doing the corresponding inverse mapping on packets on the way back. The eavesdropper only knows the internal representation of those identifiers and cannot relay the necessary information to the external collaborator. Any spoofed response from the external collaborator will be transformed to have an identifier that will result in the response getting dropped by the victim, making the attack ineffective.

The mapping can be done using either a hash function, or a state table, and is robust as long as the mapping is unpredictable. In the case of hashing, we need to use a keyed hash, with the key being the destination IP address, to prevent the attacker from using a third-party DNS server to map out the key space. The choice between state table and hash function is not always clear, as it involves space-time tradeoffs. If the hardware provides cheap hashing, then it may be preferred. In our Linksys OpenWRT implementation the use of a state table was more efficient as hashing introduced a high per-packet cost that turned the technique into a bottleneck.

5.2.4 802.11-level spoofing attack

As discussed in the context of wifi IPS, a sophisticated attacker can circumvent the ingress filtering defense by violating the 802.11 protocol to transmit frames directly to the victim. The AP can detect this by monitoring for transmissions that it did not send. However, it cannot detect the whisper attack discussed earlier, where the attacker tunes the wifi radio power so that the spoofed packets can reach the victim, but cannot reach the AP (or external detection device).

When filtering fails, the next best option is to detect and forcibly abort the attack. We pursue this direction in

the next section.

5.2.5 Whisper attack detection

We have developed a set of defenses based on the detection of abnormal combinations of network events. For example, to detect the injection of a DNS reply, we use bookkeeping of request-reply pairs to flag excess, inconsistent replies. We also raise an alert when a host appears to retransmit requests after having received replies, so we can prevent a situation where the attacker keeps inserting a fake request, just before the legitimate reply for the previous request arrives, in order to maintain the request-response balance.

While there are no visible duplicate replies in case of a whisper attack, the AP may still detect the attack indirectly. A solution for HTTP is to extract from each HTTP connection the server hostname from the corresponding mandatory HTTP header and the server address from the IP header, and compare this pair against the hostname and IP pairs extracted from observed DNS replies. If a reply has been whispered, no DNS reply will match the HTTP header and the attack will be detected.

We have evaluated our technique for detecting whisper attacks against 41,426 DNS and 339,317 HTTP requests generated by 65 IP addresses over a period of a week. We obtained 18 alerts (6 unique web sites), all of them false, corresponding to a false positive rate of 0.53×10^{-4} of all HTTP requests. For the same trace we observed zero excess DNS replies. We further evaluated only our first technique for detecting excess DNS replies against 43,272,448 DNS requests obtained over a period of more than 1 month by instrumenting an enterprise network with about 400 users. We obtained 22 alerts, all of them false, corresponding to a false positive rate of 0.5×10^{-7}. Looking deeper into the alerts revealed a Content Delivery Network that is employing spoofing, probably for server selection.

Once an attack is detected, it has to be blocked. However, given two inconsistent DNS responses, the detector cannot directly distinguish which one is legitimate and which one is spoofed. Doing a secondary lookup is one option, but the wide use of load balancing, particularly for popular services, implies that the secondary lookup may not always agree with one of the two inconsistent responses. A more relaxed check against the network prefix is also unlikely to help in the general case, as server replicas may not be co-located. In lack of any other satisfactory solution, our current implementation blocks the victim and redirects him to a warning page, notifying him of a potential spoofing attack, and giving him the option to proceed (and re-issue the request) through temporary HTTP redirection.

6 Related work

With the growing popularity of mobile devices, malware targeting wireless environment have started to emerge [27, 29]. This new security challenge has recently gained some attention from the research community.

A study related to ours is the one by Tsow et al [55]. The authors suggest that attackers could drive around a city taking over vulnerable wireless home routers. Similar to our study, the threat is amplified by dense wifi deployment, as attackers can take over hosts at a higher rate. However, the attack depends on vulnerable access points, and requires the physical presence of the attacker for driving around to find vulnerable routers. The attacks we discuss in this paper can all be launched remotely, and therefore easier and less risky for the attacker.

Anderson *et al.* [14] analyzed the speed of worm contagion over campus-wide wireless networks. They developed a worm simulation using real data from Crawdad, *e.g.* user distribution, AP distribution and user mobility, to realistically study the dynamics of a mobile worm. However their results are constrained to dynamics of mobile worm at relatively small scale of a university campus with mobility as the major factor for worm spread. In contrast, our work has investigated big cities and metropolitan areas at much larger scale with wardriving data around. We have identified a much larger threat *e.g.* infection completion in the order of minutes whereas Anderson *at al.* [14] predict a few hours to infect just the campus. The main difference is that wildfire-like propagation—not just user mobility, is the key attack vector in our work. It is also unclear whether their defense proposals could be proven effective given recent major changes of wifi usage pattern.

Beyah *et al.* [19] discuss a worm that spreads by infecting users sharing the same hotspot. They use epidemic models to simulate its spread and find it can infect a million users worldwide over the course of a year. Again the main difference is that the simulated worm relies on user mobility, but we show using wardriving data that mere density is sufficient in metropolitan areas leading to much faster spread.

Su *et al.* [52] investigate worm infections in a bluetooth environment. They expect Bluetooth to outnumber wifi devices by a factor of 5 and predict large scale epidemics, but the short range of bluetooth again implies slower, mobility-based spread. Cole *et al.* [25] use epidemic models and simulations to discuss requirements for worm mitigation in tactical battlefield MANETS.

Stamm *et al.* [49] discuss remote attacks on routers that can be used for large-scale pharming and can also spread viraly. We, too, discuss pharming as one of the potential abuses of dense, weak wifi deployments – exploitable in a different way but to a similar extent.

Mickens and Noble [43] propose a framework called *probabilistic queuing* to model the epidemic spreading in mobile environment, which aims to treat node mobility as top priority. Their simulations showed that the probabilistic queuing model could achieve more accurate prediction than standard Kephart-White framework in many cases. However, this work assumes random waypoint model for user movement and does not take into account realistic user mobility patterns.

Henderson *et al.* [33] analyzed extensive network traces from mature corporate WLANs and various university campuses and observed dramatic changes in wireless usage. Indeed, all these changes are favorable for the spread of a wifi worm. First, users now run a wide variety of applications such as peer-to-peer, multimedia and VoIP services, instead of the dominance of web traffic so there are higher chances of a worm exploitable vulnerability. Local traffic in the WLAN exceeds remote traffic, i.e. users within the same organization exchange data more than before. This would help the worm to detect and probe all wireless neighbors within its reach. The study also shows that wireless users are also surprisingly non mobile, half of which remain at home for 98% of time.

In a similar approach, Hsu and Helmy [34] found that there exists a preference of wireless user association: most users only visit a small portion of access points, i.e. the ratio of visited access points hardly changes even though popularity of WLANs increases by years – this is invariant user characteristics. There is a repetitive pattern of user association over days, i.e. there is a high probability that a user reappears at the same access point at a certain time every day. This is quantified as "network similarity index". Therefore a mobile worm could distinguish itself from traditional internet worm by self-activating at the time where most mobile users are active. This is also contrary to the general assumption and over-simplification that users are always ON with no preference on association patterns; conventional randomly generated synthetic mobility models are insufficient. Another recent trend is that a mobile node stays online on average 87.68% of its life (i.e. its existence in the wireless network). That is to say, people now tend to use WLAN as a replacement for wired network and keep their laptops constantly connected (instead of old style of establishing only when needed). A modern paradigm shift from WLAN as temporary connection to always-on permanent connection. Macro mobility: users have small converage in all environments (campus + corporate): typically only associate with 1.1% to 4.52% of total APs in their corporation. Each user has very few APs where it spends most of its online time.

Blinn *et al.* [21] monitored five weeks of Verizon wifi hotspot network in Manhattan. They observed that far more cards associated to the network than logged into it. Most clients used the network infrequently and visited only few APs. Therefore hotspot are "locations visited occasionally" rather than "primary places of work".

Kim *et al.* [37] extracted a mobility model from real user traces. Speed and pause time follow log-normal distribution and direction of movements closely related to road directions. Again, most of laptop clients are NOT very mobile, so this paper relied on VoIP users to extract mobility model. The type of mobile device being used can influence its user's mobility: a laptop would tend to tie its user to his workplace whereas a PDA/VoIP user would move as he would normally. The reasons could be due to weight, size and nature of use of the device. A mobility model for laptop users should reflect relative weightage of immobility and mobility.

Staniford *et al.* [51] describe the risk to the Internet due to the ability of attackers to quickly gain control of vast numbers of hosts. They argue that controlling a million hosts can have catastrophic results because of the potential to launch distributed denial of service (DDoS) attacks and access any sensitive information that is present on those hosts. Their analysis shows how quickly attackers can compromise hosts using "dumb" worms and how "better" worms can spread even faster. In subsequent work [50], the same authors show how a worm using precompiled lists of IP addresses known to be vulnerable can infect one million hosts in half a second. They also envision a Cyber "Center for Disease Control" (CCDC) for identifying outbreaks, rapidly analyzing pathogens, fighting the infection, and proactively devising methods of detecting and resisting future attacks. The metropolitan wifi environment offers another opportunity for attacks to occur that may not be covered by defenses built for Internet worms. Our work also provides estimates of propagation speed similar to the above studies.

The issue of location privacy in a wireless setting has been examined in literature [35, 16, 31]. These system focus attention on protecting physical location privacy based against signal triangulation techniques and protecting against source location in sensor networks. More closely related to our work, is the work by Gruteser et al [32]. The authors introduce the idea of short-lived disposable MAC addresses as a technique for the reduction of the effectiveness of location tracking. Our work shows that even in the presence of such techniques, user profiling can effectively track users in dense urban environments. Saponas *et al.* [47] describe a prototype surveillance system that can track people wearing the widely available Nike+iPod sensors. Tracknets could be exploited in similar scenarios to track people carrying any type of device whose traffic can be observed by wifi receivers, such as wifi-enabled smart-phones.

7 Concluding remarks

The increasing use of wireless technology and particularly wifi is likely to soon attract the attention of attackers, as attackers evolve and explore ways to exploit new technology to their advantage. This paper discusses a range of "modern" threats specifically tailored to metro-area wireless networks: wildfire worms that spread topologically due to infected hosts being able to carry the worm from one wireless LAN to another; large-scale wireless spoofing attacks that can be highly effective for phishing and spam campaigns; and malicious Tracknets that profile and track the whereabouts of wifi users. Such threats are greatly amplified by the increasingly dense deployment of wifi Access Points, and by the limited use of wireless security mechanisms such as 802.11i. Our results suggest that the density of large metropolitan areas has a profound impact on the severity of the threat.

Some specific contributions of this work include the modeling of fast, proximity-based worm propagation in metropolitan areas using real data from wardriving maps, wifi worm propagation using browser vulnerabilities, retrofitting of reactive mechanisms for wireless worm detection, spoofing defenses that are easy to implement, discussion of the whisper attack and defenses, and using RSS feeds to track users.

Our primary intention with this study is to raise awareness on the threats of wireless networks, specifically in densely populated areas, and to explore possible countermeasures. Much of the problem lies in the limited use of 802.11i. The wider deployment of 802.11i would reduce the risks significantly, but it would not completely eliminate them. More specifically, it would counter several instances of the spoofing threat; but it would only slow down, rather than mitigate wildfire worms; and it would not by itself eliminate the Tracknet threat, as MAC addresses remain unencrypted in 802.11i and other means of profiling may be possible.

Perhaps one of the main reasons behind the limited adoption of 802.11i is poor usability, as it involves configuration, and, once again, burdening users with yet another set of passwords or keys. Wider adoption requires convincing users that the extra trouble is worth it, by raising awareness on the risks of keeping wireless LANs open and unencrypted. We hope that our study contributes to this cause.

Improving usability of wireless security standards, if feasible, is another path to improving adoption, but until such adoption is achieved and to counter the remaining threats, we have also suggested a variety of countermeasures, which we have implemented and evaluated experimentally. Users may want to guard themselves against threats such as those described here, without having to take the cost of closing down their network using 802.11i or WEP.

Acknowledgments

We thank S.P.T. Krishnan for assisting with the vulnerability exposure analysis, and K. Xinidis for his implementation of the basic OpenWRT-based defense infrastructure. We also thank Jonathan M. Smith, Pat Lincoln, Phil Porras, Angelos Keromytis, Michalis Polychronakis, Michael Nguyen and Lee Han Boon for extremely valuable discussions and feedback on this effort.

References

[1] Aircrack: a set of tools for auditing wireless networks. `http://freshmeat.net/projects/aircrack/`.

[2] DShield.org, Distributed Intrusion Detection System. `http://www.dshield.org`.

[3] Honeynet project. `http://www.honeynet.org`.

[4] National vulnerability database, united states. `http://nvd.nist.gov`.

[5] OpenWRT Project. `http://openwrt.org/`.

[6] Securityfocus.com, vulnerabilities. `http://www.securityfocus.com/vulnerabilities`.

[7] UMich wireless usage. `http://www.itcom.itd.umich.edu/wireless/stats/yr2006/02/campus.html`.

[8] WEP: Dead Again. `http://www.securityfocus.com/infocus/1814#aircrack`.

[9] Wifitap: proof of concept for communication over WiFi networks using traffic injection. `http://sid.rstack.org/index.php/Wifitap_EN`.

[10] Wireless Geographic Logging Engine. `http://www.wigle.net/`.

[11] A year of bugs. `http://bcheck.scanit.be/bcheck/page.php?name=STATS2004&page=3`.

[12] The Spread of the Sapphire/Slammer Worm. `http://www.caida.org/publications/papers/2003/sapphire/sapphire.html`, February 2003.

[13] WMF exploitation. `http://www.f-secure.com/weblog/archives/archive-122005.html`, Dec. 2005.

[14] E. Anderson, K. Eustice, S. Markstrum, M. Hanson, and P. Reiher. Mobile contagions: Simulation of infection and disease. In *Symposium on Measurement, Modeling, and Simulation of Malware*, June 2005.

[15] W. A. Arbaugh, N. Shankar, and Y. J. Wan. Your 802.11 wireless network has no clothes. In *IEEE Wireless Communications*, 2001.

[16] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 775–784, 2000.

[17] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson. The Internet Motion Sensor: A Distributed Blackhole Monitoring System. In *Proceedings of the* 12^{th} *ISOC Symposium on Network and Distributed Systems Security (SNDSS)*, pages 167–179, February 2005.

[18] R. Beverly and S. Bauer. The spoofer project: Inferring the extent of source address filtering on the internet. In *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI) Workshop*, pages 53–59, July 2005.

[19] R. A. Beyah, C. L. Corbett, and J. A. Copeland. The case for collaborative distributed wireless intrusion detection systems. In *IEEE International Conference on Granular Computing*, May 2006.

[20] A. Bittau, M. Handley, and J. Lackey. The final nail in wep's coffin. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 386–400, Washington, DC, USA, 2006. IEEE Computer Society.

[21] D. P. Blinn, T. Henderson, and D. Kotz. Analysis of a Wi-Fi hotspot network. In *Proceedings of the International Workshop on Wireless Traffic Measurements and Modeling*, June 2005.

[22] N. Borisov, I. Goldberg, and D.Wagner. Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of ACM Mobicom, Rome, Italy*, July 2001.

[23] S. Byers, L. F. Cranor, D. P. Kormann, and P. D. McDaniel. Searching for privacy: Design and implementation of a P2P-enabled search engine. In D. Martin and A. Serjantov, editors, *Privacy Enhancing Technologies*, volume 3424 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 2004.

[24] J. Cache and D. Maynor. Device drivers. Presentation at Blackhat USA 2006, August 2006.

[25] R. G. Cole, N. Phamdo, M. A. Rajab, and A. Terzis. Requirements on worm mitigation technologies in MANETS. In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, pages 207–214, Washington, DC, USA, 2005. IEEE Computer Society.

[26] T. Espiner. Does Wi-Fi security matter? `http://news.com.com/2100-1029_3-6088741.html`.

[27] F-Secure. Cabir worm description. `http://www.f-secure.com/v-descs/cabir.shtml`, June 2004.

[28] F-Secure. Data Security Summary - January to June 2005. `http://www.f-secure.com/2005/1/data-security-summary-2005_1.pdf`, 2005.

[29] F-Secure. Inqtana.A worm information. `http://www.f-secure.com/v-descs/inqtana_a.shtml`, Feb. 2006.

[30] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Lecture Notes in Computer Science, 2259:124*, 2001.

[31] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 620–629, Washington, DC, USA, 2005. IEEE Computer Society.

[32] M. Gruteser and D. Grunwald. Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. *Mob. Netw. Appl.*, 10(3):315–325, 2005.

[33] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking (MobiCom)*, Sept. 2004.

[34] W.-J. Hsu and A. Helmy. On modeling user associations in wireless LAN traces on university campuses. In *Proceedings of the Second Workshop on Wireless Network Measurements (WiNMee 2006)*, Apr. 2006.

[35] Hu and Wang. Framework for location privacy in wireless networks. In *ACM SIGCOMM Asia Workshop*, 2005.

[36] M. Hypponen. WLAN viruses, anyone? F-Secure weblog. `http://www.f-secure.com/weblog/archives/archive-082006.html`, August 2006.

[37] M. Kim, D. Kotz, and S. Kim. Extracting a mobility model from real user traces. In *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Apr. 2006.

[38] B. Krebs. 2005 patch times for Firefox and Internet Explorer. Washington Post weblog. `http://blog.washingtonpost.com/securityfix/2006/02/2005_patch_times_for%_firefox_a.html`, Feb. 2006.

[39] B. Krebs. Internet Explorer unsafe for 284 days in 2006. Washington Post weblog. `http://blog.washingtonpost.com/securityfix/2007/01/internet_explorer_un%safe_for_2.html`, Jan. 2007.

[40] N. Leavitt. Mobile Phones: The Next Frontier for Hackers? *IEEE Computer*, 38(4), April 2005.

[41] C. Martel and V. Nguyen. Analyzing Kleinberg's (and other) small-world models. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 179–188, New York, NY, USA, 2004. ACM Press.

[42] R. McMillan. Researchers hack wi-fi driver to breach laptop. InfoWorld. `http://www.infoworld.com/article/06/06/21/79536_HNwifibreach_1.html`, June 2006. Accessed on September 15th, 2006.

[43] J. Mickens and B. Noble. Modeling epidemic spreading in mobile environments. In *Proceedings of the ACM Workshop on Wireless Security*, pages 77–96, 2005.

[44] S. Milgram. The small world problem. In *Psychology Today*, 1967.

[45] G. Portokalidis, A. Slowinska, and H. Bos. Argos: an emulator for fingerprinting zero-day attacks. In *Proc. ACM SIGOPS EUROSYS'2006*, Leuven, Belgium, April 2006.

[46] M. Roesch. Snort: Lightweight intrusion detection for networks. In *Proceedings of USENIX LISA*, November 1999. (software available from *http://www.snort.org/*).

[47] T. S. Saponas, J. Lester, C. Hartung, and T. Kohno. Devices That Tell On You: The Nike+iPod Sport Kit. Technical report, Department of Computer Science and Engineering, University of Washington, 2007.

[48] C. Shannon and D. Moore. The Spread of the Witty Worm. *IEEE Security & Privacy*, 2(4):46–50, July/August 2004.

[49] S. Stamm, Z. Ramzan, and M. Jakobsson. Drive-By Pharming. Technical report, Indiana University, Dec. 2006.

[50] S. Staniford, D. Moore, V. Paxson, and N. Weaver. The Top Speed of Flash Worms. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, pages 33–42, October 2004.

[51] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in Your Spare Time. In *Proceedings of the USENIX Security Symposium*, pages 149–167, August 2002.

[52] J. Su, K. K. W. Chan, A. G. Miklas, K. Po, A. Akhavan, S. Saroiu, E. de Lara, and A. Goel. A preliminary investigation of worm infections in a bluetooth environment. In *WORM '06: Proceedings of the 4th ACM workshop on Recurring malcode*, pages 9–16, New York, NY, USA, 2006. ACM Press.

[53] Symantec. Symantec Internet Security Threat Report. `http://www.symantec.com/enterprise/threatreport/index.jsp`, Sept. 2006.

[54] Trend Micro. Botnet threats and solutions: Phishing - Whitepaper, Nov. 2006.

[55] A. Tsow, M. Jakobsson, L. Yang, and S. Wetzel. Warkitting: the drive-by subversion of wireless home routers. *Anti-Phishing and Online Fraud, Part II Journal of Digital Forensic Practice*, 1(3), Nov. 2006.