Mobile Evil Twin Malnets - The Worst of Both Worlds

Christian Szongott, Benjamin Henne, and Matthew Smith

Distributed Computing & Security Group
Gottfried Wilhelm Leibniz University of Hannover
{szongott,henne,smith}@dcsec.uni-hannover.de
http://www.dcsec.uni-hannover.de

Abstract

The mobile computing world is undergoing major changes both in the capability as well as in the proliferation of mobile devices. While, up to now, mobile malware has played a relatively small role compared to the behemoth of desktop malware, the changing environment is steadily increasing the attractiveness of mobile devices as exploitable resources. The increased usage and connectivity of mobile devices opens up a much larger set of attack vectors to compromise them. In this paper, we adapt the evil twin rogue access point attack to the mobile domain and show how it can be used to create a mobile malnet, which is capable of spreading epidemically. We implemented the key components of the concept for the iPhone to study its properties in a laboratory environment. To demonstrate the dangers which come along with this kind of attack we simulate a metropolitan area and show how fast a malware can spread in a mobile environment.

1 Introduction

Researchers have predicted the epidemic spread of mobile malware many times in the last decade. However, a true outbreak of mobile malware has not occurred in the wild yet, leading to a certain level of complacency towards the threat of mobile malware.

That mobile devices have not been targeted in the past is partly due to the fact that the proliferation and capabilities of mobile devices were so limited that they were consequently also only of limited use to attackers. Compared to the effectiveness of Wi-Fi desktop worms with epidemic qualities such as the wildfire worm presented by Akritidis et al. [1] the capabilities of worms for mobile phones have been relatively small.

However, several key factors in mobile computing have changed in recent years, significantly altering the playing field and giving rise to a new threat of mobile malware [2]. The proliferation and capabilities of mobile, networked devices has increased markedly and the mobile phone has become the central digital hub of our lives, storing personal information, multimedia content as well

as offering access to social networks, online banking, work networks and a host of other functionality. This increases both the value as a target as well as the attack surface. Recent work has also shown the potential dangers of mobile malnets [3]. Malnets are botnets created from routers, cellphones, and other non-traditional computational Wi-Fi devices. The main focus of malnet research has been in the area of router attacks, such as presented by McDaniel [4].

In this paper we show how recent advances in mobile devices can be used to combine the most dangerous aspects of these two worlds - mobile malnets and Wi-Fi worms. We will show how the inclusion of mobile hotspot capabilities in virtually all new mobile devices opens the door to leverage the evil twin attack[1] to create a mobile malnet capable of spreading from device to device. Unlike in previous work we do not require vulnerabilities in the Wi-Fi access points, nor it is a problem that most private and corporate Wi-Fi networks now use WPA to encrypt their traffic. To test the feasibility of our approach we implemented the key components needed to create the malnet for iOS. In lab experiments we analyze the basic parameters and requirements of this malware mechanism and measure infection and propagation times in a controlled environment.

The rest of the paper is organized as follows: In section 2 related work in the field of mobile malware and evil twins is presented. Section 3 describes how mobile evil twin malnets work and which weaknesses of todays smartphones are utilized to render this type of attack possible. Additionally our prototypical implementation is described in-depth. In Section 4 we present the simulations we used to study the spreading characteristics, their assumptions and results. Section 5 concludes the paper and gives an overview of possible future work in this research area.

2 Related Work

There is a large body of related work discussing the spread of mobile device malware. Most of this work [5–11] relies on Bluetooth or user errors as an infection vector. One prominent example for Bluetooth-based worm research is described in a paper from Wang et al. [12] where they analyzed the spreading patterns of mobile viruses. They considered Bluetooth and multimedia messaging as distribution channels for this kind of worms. They predict serious threats to mobile phones once an operating system reaches high enough market shares. How vulnerable even *feature phones*[2] are, is shown in the work by Collin et al. [11]. They studied the vulnerabilities of these mobile phones and possible attacks against the mobile network using SMS and Bluetooth.

In 2007 Akritidis et al. [1] presented a simulated study of Wi-Fi-based malware in metropolitan area networks. The work explores the idea to utilize the proximity of Wi-Fi routers to spread malware. Akritidis et al. present two modes

[1] Evil twin is a term for a malicious Wi-Fi access point that appears to be a legitimate one by spoofing its SSID

[2] Phones that are not considered to be smartphones, but have additional functionality beyond standard mobile services

of infection labeled push and pull. Push infection resembles traditional worm infection methods most closely. With push infection a vulnerability in a device connected to a vulnerable hotspot is exploited to infect that device, which then goes on to infect further devices. The second method is the pull method. In this case the infected node waits for a device connected to a vulnerable access point to make some form of network request and then injects the worm code into this connection. The advantage of this method is that instead of relying on a service vulnerability, the attacker exploits vulnerabilities, such as browser vulnerabilities which are much more frequent and patched at a slower rate. In the studies case both the push and the pull method rely on visible, open and unencrypted access points to launch the attacks against other connected devices.

Another source of related work and inspiration for our work are malnets. Malnets are botnets created from non-traditional Wi-Fi devices such as mobile phones, but in particular wireless routers. In their work McDaniel [4], Hu et al. [13] and Traynor et al. [14] show that traditional network routers can be used to build up large malnets in areas with a high density of routers. The attack uses weak or default passwords to compromise wireless routers, which are in range, which then in turn attempt to infect other vulnerable routers in range. They developed epidemiological models and showed that in densely populated areas with enough vulnerable routers an epidemic spread of the malnet is completed within hours. While not strictly being a malnet the research presented by Akritidis et al. above has similar properties. But since these works only consider static environments where APs themselves have to be infected we describe in this work a new kind of infection and explore an alternative route to creating malnets, which does not require vulnerable routers.

The attack vector we utilize is based on the evil twin attack presented by Bauer et al. [15]. They showed that it is possible to trick a wireless client into associating with a rouge (evil twin) access point. The paper describes how an attacker can execute MITM attacks against selected victims by tricking them into connecting with the evil twin hotspot instead of the legitimate one. While this was viable and serious even then, it did not receive a great deal of attention. We assume this lack of interest was due to the fact that at the time of the attack it was only discussed in the context of infrastructure-based Wi-Fis, where targeted attacks against specific locations were the goal. The proliferation of WEP, WPA/WPA2 made this attack significantly harder in most scenarios. Also the effort of setting up the evil twin and the targeted nature of the attack reduced the mass-effect of the approach. The changes in the mobile computing landscape and our research into extending this attack by leveraging the capabilities of current smart mobile devices to create a potentially massive mobile malnet increases the threat significantly.

3 Mobile Evil Twin Malnets

In this section we show that the currently dormant threats shown above can be combined and extended to once again be a serious threat in the new mobile

computing landscape. There are two key factors which enable the approach presented in this paper; the proliferation of Universal Authentication Mechanism (UAM) Wi-Fi hotspots [16] accessed through captive portals and the capability of modern smart mobile devices to act as a mobile hotspot. Before we present the concept of the mobile malnet, we discuss the usage and security environment which has now made this type of malware possible.

3.1 Wi-Fi Hotspot Usage and Security

Wi-Fi access has become a commodity service in most urban areas, such as malls, coffee shops, hotels, airports and other public locations. A relatively small number of Wireless Internet Service Providers (WISPs) offer Wi-Fi access for either ad-hoc usage or bundled with other services such as mobile phone plans.

While most Wi-Fi routers now usually come pre-configured with WPA/WPA2 or similar channel encryption mechanisms, public Wi-Fi is often unencrypted. The lack of channel encryption for public Wi-Fi is mostly due to the fact that WISPs have not found a way to incorporate the necessary configuration steps into their business and usage model. While the lack of encryption brings with it a number of security problems such as impersonation, credential theft and other violations of a Wi-Fi user's privacy, the lack of strong mutual authentication during the connection to a public Wi-Fi hotspot and the unsecured use of SSIDs (Service Set Identifiers) for both ESS (Extended Service Set) and IBSS (Independent Basic Service Set) networks is the main problem.

In most public Wi-Fis authentication of users is handled by the UAM. With UAM, any device is allowed to associate with the Wi-Fi access point (AP) and is issued an IP address and other network information such as the standard gateway automatically via DHCP. After association, the user opens a web browser and enters any URL. A transparent HTTP proxy (also called captive portal) on the AP (or the underlying infrastructure) captures the request and redirects it to a login page. In the case of free Wi-Fi, the user is usually just required to accept the terms of use; in the case of subscription or pay-as-you-go Wi-Fi the user needs to provide valid credentials or purchase access on the fly. The credentials entered in the login form are forwarded to a back-end processing facility (AAA server, credit card processor, etc.) and after successful validation the user's session is started. Now the user's primary HTTP request is sent and the response is delivered to the user. Critically, the content of this portal is controlled entirely by the WISP. Later in this paper we will show that this issue creates several dangerous problems.

Today there is no practical way to authenticate APs wherefore it is neither mandatory nor automated and common. If at all, it needs to be performed by the user checking a SSL/TLS certificate. Unlike the use of certificates in traditional client/server Internet applications (which also can be fraught with difficulty), this measure has very little security effect in practice, when applied to hotspot security. SSL certificates are issued for a FQDN (fully qualified domain name). However, since there is no verifiable or even any form of specified link between the FQDN and the SSID, the certificate-based approach is not feasible. Thus,

the vast majority of users will not be able to associate a hotspot SSID with the "correct" host name for the hotspot's authentication service.

The final problem in current Wi-Fi security is the fact that for increased usability many mobile devices are configured to automatically connect to known Wi-Fi hotspots, where "known hotspot" is defined only by the SSIDs being equal. Since SSIDs are not authenticated this opens up the possibility of the evil twin attack [17].

3.2 Mobile Evil Twin Attack

If a device has been connected to a hotspot with a given SSID at any time in the past, it is susceptible to an "evil twin" attack. An attacker positioned outside the Wi-Fi range of the real hotspot or with a stronger signal can create an evil twin hotspot by spoofing its SSID. In the previous section, we showed that there is currently no practical way for users of a UAM-enabled Wi-Fi hotspot to reliably determine its authenticity. So if an evil twin broadcasts a "known" SSID and a device connects to it, the evil twin is then capable of executing man in the middle attacks (MITMAs) against that device. In this paper we introduce an adaptation of the evil twin attack to create a far more potent attack type, the Mobile Evil Twin attack (MET). Unlike the evil twin attack, the MET does not aim at subverting a specific AP to carry out targeted attacks, but to misuse the UAM weaknesses as an infection channel and replicate itself.

In order to spread, any malware needs to find vulnerable hosts and an infection vector. The environment described above offers both. In a lot of countries, mobile hotspots are ubiquitous in many establishments like Starbucks, Barnes&Noble and McDonald's. These hotspots usually have a single SSID per brand to allow customers to easily connect to hotspots across any of their locations. In some cases even the WISP's default SSID is used (i.e. by BT in the US and the UK, which use the SSID `BTOpenzone`). While the default SSID for public Wi-Fi makes sense from a usability perspective, it opens the door to the MET attack since it creates a large user base which has connected to a well known SSID. In the following, we will show the design of the MET attack and the prototypical implementation of the key components. The implementation extends the jailbreakme.com jailbreak to deploy all components for a self-replicating mobile malnet. All components were separately tested in the lab using iOS 4.3.3 running on an iPhone 4, an iPad 1 and an iPad 2. The principle can be ported to any OS with a root vulnerability. However, the usability feature of an automatic browser-based captive portal popup makes iOS the more attractive target. The whole attack process consists of four steps that are graphed in Figure 1 and described below.

Step 1: Masquerading & Injection The first step of the MET attack is the exploitation of the auto-reconnection feature to known hotspots. The MET attack starts with the initial infection host broadcasting an SSID corresponding to a well used SSID such as those used by public hotspots like *tmobile*. If a mobile

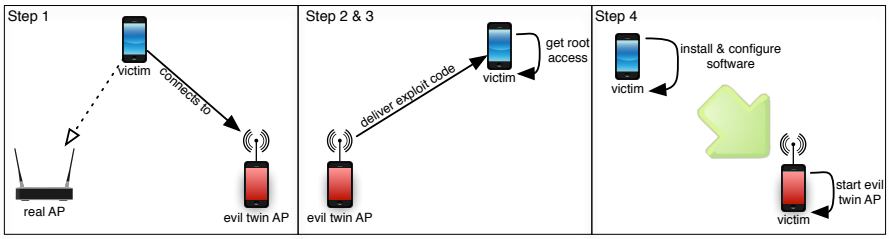


Fig. 1. MET attack overview

device has previously been connected to a legitimate AP of that SSID label, it is likely that it will automatically connect to the MET hotspot, unless there is a competing AP with higher signal strength or a SSID with higher priority.

Once the mobile device is connected to the MET hotspot all Internet activity of the device goes via the hotspot. With this setup, we can run a rogue Wi-Fi hotspot which can not only sniff, but also modify all unencrypted communication sent through it and inject malicious content.

The proof of concept malware presented in this paper uses `pf`, the iOS internal firewall and packet forwarding engine, to redirect all incoming traffic on port 80 to a locally deployed web server where the malicious payload is hosted. A more detailed description will follow later in this paper.

Step 2: Application Exploit Once the mobile device is connected to the MET hotspot the second step of infection can occur. Since we use iOS for the prototypical implementation, which creates a MobileSafari popup window during the login process on the captive portal we can utilize a browser-based attack.

For all of the relevant smartphone platforms, i.e. iOS, Android, Palm/HP WebOS, BlackBerry and Symbian, the preinstalled web browsers are based on the WebKit [18] rendering engine. The WebKit engine is under constant scrutiny by security researchers and has had a number of security issues in the past (e.g.. [19–21]). Many of these security issues could be exploited by an attacker to execute arbitrary code on the phone with the same permissions as the web browser. Based on the long years of experience with desktop browsers and the recent history of mobile browsers, it can be assumed that mobile browsers will continue to be affected by security issues that allow arbitrary code execution for a fair while. For the proof of concept implementation we used the jailbreakme.com jailbreak which makes use of a PDF rendering vulnerability of the CoreGraphics framework in iOS 4.3.3 [22, 23].

Step 3: Kernel Exploit Executing code with the browser's permissions is rarely sufficient to install malware on a device. Compartmentalization and "sandboxing" of processes is a typical security approach for modern operating systems, including those on smartphones. Usually this means that the web browser –

which runs under an unprivileged user id – does not have privileges to install software on the phone. Therefore, another successful exploit on the operating system level is necessary to obtain full administrative privileges. Although harder to find than bugs in WebKit, these exploits exist and have existed for most mobile operating systems and it is likely that they will be available for future OS versions.

Step 4: Crossing-over into the Evil Twin Malnet Once a kernel-level exploit was executed on a mobile device the attacker gains complete control over that device, including the ability to control network and radio functions. This control is necessary for the propagation concept which we will outline in the following section. It makes use of the "personal hotspot" feature (Apple iOS $\geq$4.3). This hotspot functionality can be used by the device owner to share their 3G Internet connection with others by creating a mobile Wi-Fi access point. It conveniently bundles a small DHCP server, routing capabilities and Wi-Fi channel management and the setup is trivially easy. During the MET attack the mobile hotspot is activated using the target SSID making the device a new MET carrier.

By deploying *lighttpd* as a light weight web server on the infected mobile phone and rerouting all HTTP traffic of connected devices to it, we can deliver malicious code within any HTTP response and thus infect more mobile devices. Now, the infection cycle starts back at step 1.

3.3 Automating Infection

In the steps above an infection only occurs when a user manually triggers an event that receives content from the Internet into which the attack code can be injected. However, iOS has a convenient usability feature which enables almost instantaneous infection. When the iOS device is active, it automatically connects to the MET captive portal and as soon as any application requires Internet access an automated pop-up meant for credential input is shown.

This login page contains content which is fully controlled by whoever runs the Wi-Fi hotspot. It is, moreover, rendered with the same browser engine that is used for all other browsing activities. Therefore, virtually all exploits and security issues that exist for the WebKit browser family can also be exploited on that login page. Since the login page is displayed automatically, the MET infection routine can be executed as soon as the iOS device requests data from the MET hotspot.

Under the hood the following happens. After successfully associating with an AP and receiving an IP address, the iPhone's network stack sends one HTTP request to a specified URL. In the majority of cases the request goes to `http://www.apple.com/library/test/success.html`.

The response is then evaluated by iOS and if it consists of a valid HTML document containing only HTML metadata and exactly the word "Success" in the title tag, the network stack uses this as an indicator for the Internet

access being functional and finishes the connection setup. If any other response is received, the `apple.com` website is requested. In a second step the corresponding response is shown to the user in an automatic full-screen pop-up window which we use to deliver the exploit. The reason for this is the Universal Authentication Mechanism (see Section 3.1) where the user is prompted for his credentials by the captive portal login page.

3.4 Infection Implementation

In the following we describe the infection process with all involved systems and configurations. In the proof of concept presented in this paper we used a jailbroken iPhone 4 running iOS 4.3.3 as the initial infecting device. All further devices did not need to be prepared or jailbroken since all jailbreaks were executed automatically by the MET. The components of the prototype were tested on iPhone 4, iPad 1 and iPad 2.

To run an evil twin access point hotspot software needs to be configured that opens up Wi-Fi networks spoofing legitimate hotspot SSIDs. As the iOS internal personal hotspot cannot be activated and configured from the command line[3], we deployed MyWi 4[4], an app for jailbroken devices that has the same features, but can be configured through a plist file. It can be started through an undocumented program parameter from a shell script. This adds 1.8 MB to the MET prototype which could be avoided, but since optimizing the malware is not in the focus of this paper we chose the convenience MyWi offers. In addition to MyWi we installed the web server lighttpd due to its easy configuration and small footprint. The server is set up to listen on port 80 and is ready to deliver the exploit and the payload.

When a victim connects to this MET hotspot he gets an IP address from the local DHCP server that comes along with MyWi. We set up the firewall rules shown in Listing 1.1 to redirect all port 80 HTTP traffic from the hotspot network to the local lighttpd server.

```
1  nat on pdp_ip0 inet
2    from 192.168.40.0/24 to any
3    -> (pdp_ip0:0) static-port
4  no nat on ap0 inet
5    from 192.168.40.1 to 192.168.40.0/24
6
7  rdr on ap0 proto tcp
8    from 192.168.40.0/24 to any port 80
9    -> 127.0.0.1 port 80
10
11 pass on pdp_ip0
12   from any to any flags S/SA keep state
13 pass on ap0 all flags any
14   xkeep state (source-track global) rtable 4
```

Listing 1.1. PF firewall rules including the redirection for incoming traffic (line 7-9)

[3] The hotspot functionality relies on undocumented iOS Kernel APIs
[4] Product website: http://intelliborn.com/mywi.html

When the victim's device checks the Internet connection, the web server responds to the first request of `http://www.apple.com/library/test/success.html` with a non-success webpage. Thus, the pop-up opens and displays the content of `http://www.apple.com`, which is also delivered by the local web server.

In this case it contains a hidden HTML iframe element showing a PDF file. Even within the popup and the hidden iframe, the iOS browser MobileSafari renders the PDF file. The malicious PDF file then exploits a vulnerability of the CoreGraphics library which is called by the Webkit engine. We use the exploit code from `http://www.jailbreakme.com` which first leads to a root shell and then uses it to download and install the actual jailbreak.

At this point there is an implementation issue with the jailbreakme.com PDF exploit. After the initial exploit is executed it tries to download the files that are required for the rest of the jailbreak and Cydia installation process. But since we are still in captive portal mode and the network state is set to "no internet connection" the download fails. Serving the success page for the portal and recapturing the device could solve this. However, the download would have to be delayed briefly until a functioning internet connection is recognized by iOS. This is not a conceptual problem and only stems from the fact that we are recycling an existing exploit which can not be modified at that point. If the malware was to be created for real a new exploit would be needed in any case and the delay would not be a problem. This issue does not affect the first infection case described above when a web browser is used. During the jailbreak process file downloads are initiated through plain HTTP requests without any authentication of the download server. Thus, we can redirect all requests to `http://www.jailbreakme.com` to the local web server as well and replace the jailbreakme.com payload with our own.

The jailbreak itself consists of three main parts: an initial filesystem image contains the directory structure including the `/bin` directory with utilities such as dpkg; a Debian package containing further programs; and a dynamic library that triggers and controls the installation of all these components. During the jailbreak process the integrity of the initial filesystem is checked. However, there is no integrity check made for the Debian package. Thus, we can modify the package and deliver it to the victim's device when the jailbreak process attempts to download the original one.

The modified Debian package contains not only the jailbreak files, all MET-related packages and configurations, a metadata file and a description of the package, but also command and control shell scripts that are executed before and after the installation or removal of the package. In the post-installation script of the modified package the installation of further packages is triggered. However, it is not possible to use the dpkg package manager directly for this purpose since the packaging system is locked during a running installation process. To circumvent this problem we register a daemon script to the iOS launch daemon `launchd` which usually monitors running iOS services and restarts them in the case of an abnormal termination (the script can be found in Listing 1.2 of appendix A).

Thus, after the post-installation script of the Debian package terminates successfully and removes the lock, iOS activates the daemon script and installs the rest of the malware. The daemon script checks if it was executed successfully in the past. If so, the daemon script deregisters itself from the iOS launchd service daemon and terminates. This is done to make sure that the installation of the evil twin software is not triggered twice. Otherwise it launches the installation of Debian packages that have been transferred to the victim's device within the Debian package. The hotspot software MyWi, the web server lighttpd and all of their package dependencies are included. The daemon script can be found in Listing 1.3 of appendix A.

To operate the malicious hotspot the two main components are started. The hotspot is started by a simple shell script call. For the web server a start script is registered as a daemon at the `launchd` service, like in the installation process. Now we have a fully operational evil twin hotspot that is ready to infect other devices.

There are currently three usability issues in the MET prototype. Firstly, as mentioned above, the popup-based installation would require a short delay to restore Internet access after the local redirection. Secondly, the installation of MyWi requires a restart of the device which would either be noticeable by the user or require the malware to wait for the next ordinary reboot. Thirdly, we do not hide the fact that we jailbreak the device. While the jailbreak process is executed in the background, we leave the AppStore equivalent Cydia in place. All three issues are implementation issues which do not affect the MET concept. Delaying the additional downloads by a couple of seconds solves the first issue. The second could be solved by subverting the iOS hotspot capability and the third would require the removal of the visible jailbreak signs. However, since it is not the goal of this work to create a stealthy real world malware and since these issues do not affect the concept of the MET attack there is currently no plan to address these issues.

4 MET Simulation

We used simulations to analyze the potential spreading characteristics of the mobile malware prototype presented in this paper. We developed a simulation framework [24] to study various security and privacy related issues in mobile environments. The framework allows us to carry out agent-based simulations on real-world road networks that are based on geospatial data from OpenStreetMap[5]. For the simulations in this work we used the framework and chose Downtown Chicago as the simulation area. Based on transport statistics we estimated a total number of 400,000 people using smartphones in downtown Chicago. Since we are simulating an exploit for the iOS platform we take the iOS market share of 20% [25] and the vulnerable iOS versions ($\sim$ 5% of iPhones are still running iOS version 4) into account. Thus, we have an infection base of about 4,000

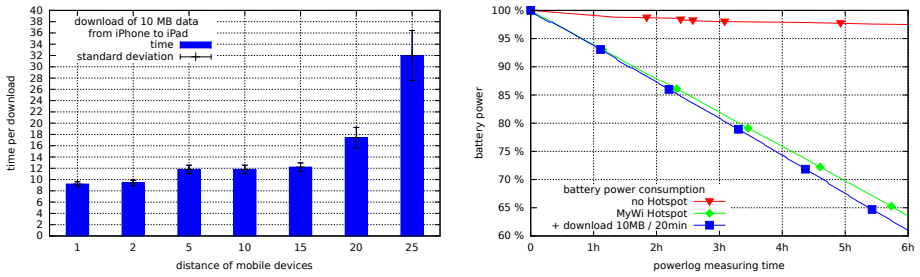
[5] `http://www.openstreetmap.org`

devices. Considering the rising iOS market share and possible future multiplatform exploits we ran simulations also with a population size of 10,000 people. We implemented several types of agents which have different behavior patterns and walking speeds as well as diverse mobile device usage patterns. An agent chooses between different actions with a specified probability depending on his type.

Crucial factors for the simulation are the transmission speed and the battery lifetime of the devices. Therefore we investigated parameters like the infection duration and battery consumption of the prototype in order to set up the simulations as realistic as possible. We also conducted several parameter studies, varying population size as well as other parameters and found out characteristics for this type of attack in urban areas. For a more detailed description of the simulations we refer to [26].

4.1 Infection Duration & Battery Consumption

We conducted several lab experiments to determine the parameters of the MET prototype with the above limitations. The PDF file containing the initial exploit has a size of 17kB and its transfer plus the execution of the initial exploit takes under 1 second. This was measured using the browser based infection vector. Following this, the rest of the malware is downloaded by the initial exploit. Since this value is of relevance for the spreading characteristics of the malware, we measured the time needed for this download. Therefore we transferred a comparable 10MB MET package between the iPhone running the MET malware and a victim iPad at different distances.



(a) MET transfer duration measured over the distance between the mobile devices

(b) Battery consumption of the MET

Fig. 2. MET-related measurements

Figure 2a shows that the download time at the range of 1-2 meters was about 9 seconds. At distances of up to 15 meters download times rise to approximately 12 seconds, increasing to more than 32 seconds at a 25-meter distance. The

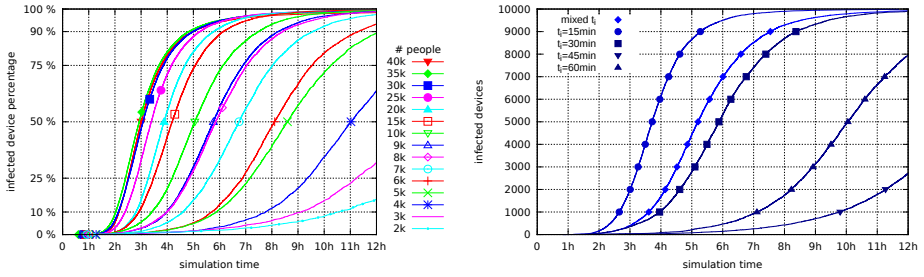
measurements were conducted outside in a populated area with four other Wi-Fi hotspots active in the immediate vicinity. Each measurement was repeated five times and the standard deviation is shown in the figure.

Figure 2b shows the mean battery consumption of the MET measured over a six hour period. The red line marked with a triangle shows the battery consumption of the uninfected phone being idle. The green line marked with a diamond represents an infected phone with an active MET hotspot but otherwise idle. The blue line marked with a square shows a MET device which infects another device every 20 minutes. As can be seen the MET, if operated continuously, might be noticeable for the user due to the shortened battery lifetime. But with this shortened battery lifetime the infection of numerous other devices is still possible. Even if a user notices a suspicious behavior of his device, he will be not able to remove the MET without a complete phone reset. However the current version of the MET has not been optimized for stealth or longevity, since it is out of the focus of this paper.

The measured values are the basis for the simulations of mobile phones with limited battery life. In addition to the regular consumption each time a device infects another one, a small amount of battery lifetime is subtracted representing the running hotspot and the transfer of the mobile malware. If a device becomes infected its battery lifetime is reduced according to the measured values. Devices that run out of battery during the simulation neither can become infected nor can infect other nearby devices.

4.2 Simulation Results

In the following we present the results of the simulations to give a rough idea of how this kind of mobile malware could spread in an urban area.



(a) Infection simulation with different population sizes

(b) Infection simulation with different internet usage intervals

Fig. 3. Parametric study of the MET attack

Since the number of infectable devices has a significant effect on the epidemic spread of the malware, we conducted a parameter study over the number of

devices in a closed world scenario, where no agent enters or leaves the simulated area. Figure 3a shows the normalized results of this study. As can be seen from 5,000 devices onwards an almost total infection occurs within 12 hours. From 10,000 devices onwards almost total infection is achieved within 8 hours.

To show the effect of usage patterns on the spread of the mobile malware, Figure 3b plots the amount of infections with different device activation intervals. As can be seen in the Figure the spreading of the malware has an epidemiological character for Internet usage intervals less than 30 minutes. Device usage intervals greater than 30 minutes lead to a significantly slower but nevertheless substantial spreading of the malware.

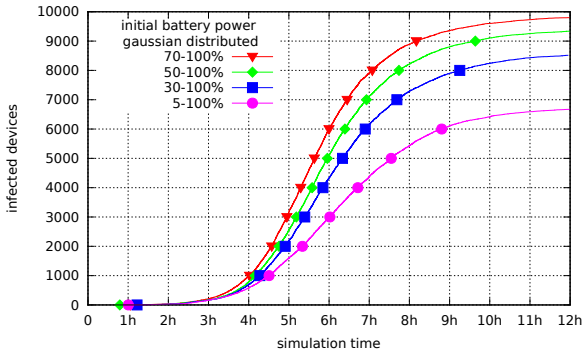


Fig. 4. Infection of 10,000 people depending on initial battery levels

Figure 4 shows a parameter study with different distributions of initial charge. Each simulation starts with a Gaussian distribution of initial charges in the range shown in the key of the figure. As can be seen the distribution of initial battery levels has a greater effect than the battery power drain. Surprisingly, even when a large number of devices run out of battery during the simulation there are still enough active infected devices to create almost full infection.

4.3 Mobile Evil Twin Malnet Summary

To summarize, the Mobile Evil Twin attack introduced above represents a new approach for creating mobile malnets. Unlike current Internet- (e.g.. spyeye [27]) or App Store-based (e.g.. Droid Dream [28]) attacks the user is not required to surf to a specific site or install a specific App to be infected. Unlike previous malnet approaches we do not require vulnerabilities in hotspots nor does the wide scale deployment of WPA/WPA2 pose a problem as long as there are some popular captive portal-based hotspots in operation somewhere. The concept for the automatic popup based infection was shown in theory and the web browser-based infection vector was fully implemented and shows the viability of the attack. The type of mobile malware presented in this work is made possible by

both personal hotspot capabilities of smart devices as well as the fact that the usability and security measures in place for using and protecting Wi-Fi devices were designed without this scale of capabilities in mind. Thus they offer very little protection against attacks in this realm. The conducted lab tests show the viability of the approach on a device-to-device basis.

5 Conclusion and Outlook

In this paper, we showed how new capabilities of mobile devices can be used to create a mobile evil twin malnet which is capable of spreading epidemically in current metropolitan areas. We present a proof of concept implementation for iOS 4.3.3 and measured the key attributes of the MET. In lab experiments we measured the infection time and battery consumption of the prototype to use these values in simulations. We simulated a metropolitan area to analyze the spreading characteristics of this new kind of mobile malware and showed that a large user base gets infected within a small amount of time.

There are many areas of future work. The mobile malware is just a proof of concept and could easily be made more effective or stealthy depending on the desired use. The malware can be extended to exploit more than one OS and cycle between different SSIDs to significantly increase the potential infection base and thus also significantly speed up the infection rate. Or the rate could be artificially slowed to make the spread more stealthy and energy saving. More importantly, an effective countermeasure needs to be researched to combat this type of malware. Current countermeasures do not work well against MET, thus opening up room for new ideas and research in this area. One promising idea is to use context information, such as the location or other environmental parameters to augment the decision when to automatically connect to known SSIDs or raise an alarm. IPSs and IDSs could also be ported to mobile platforms to collaboratively detect emerging malnets.

References

1. Akritidis, P., Chin, W.Y., Lam, V.T., Sidiroglou, S., Anagnostakis, K.G.: Proximity Breeds Danger : Emerging Threats in Metro-area Wireless Networks. In: Proceedings of 16th USENIX Security Symposium. (2007) 22:1–22:16
2. Becher, M., Freiling, F.C., Hoffmann, J., Holz, T., Uellenbeck, S., Wolf, C.: Mobile security catching up? revealing the nuts and bolts of the security of mobile devices. In: 2011 IEEE Symposium on Security and Privacy. (May 2011)
3. Husted, N., Myers, S.: Mobile location tracking in metro areas: malnets and others. In: Proceedings of the 17th ACM conference on Computer and communications security, ACM (2010) 85–96
4. McDaniel, P.: Malnets: large-scale malicious networks via compromised wireless access points. Security and Communication Networks (2009)
5. Khouzani, M.H.R., Sarkar, S., Altman, E.: Maximum damage malware attack in mobile wireless networks. In: Proceedings of the 29th conference on Information communications. INFOCOM'10, IEEE Press (2010) 749–757

6. Yan, G., Flores, H.D., Cuellar, L., Hengartner, N., Eidenbenz, S., Vu, V.: Bluetooth worm propagation. In: Proceedings of the 2nd ACM symposium on Information, computer and communications security - ASIACCS '07. (2007) 32
7. Singh, K., Sangal, S., Jain, N., Traynor, P., Lee, W.: Evaluating Bluetooth as a medium for botnet command and control. In: Proceedings of the 7th international conference on Detection of intrusions and malware, and vulnerability assessment, Berlin, Springer-Verlag (2010) 61–80
8. Fleizach, C., Liljenstam, M., Johansson, P., Voelker, G.M., Mehes, A.: Can you infect me now? In: Proceedings of the 2007 ACM workshop on Recurring malcode. WORM '07 (2007) 61
9. Dagon, D., Martin, T., Starner, T.: Mobile Phones as Computing Devices: The Viruses are Coming! IEEE Pervasive Computing **3**(4) (October 2004) 11–15
10. Carettoni, L., Merloni, C., Zanero, S.: Studying Bluetooth Malware Propagation: The BlueBag Project. IEEE Security and Privacy Magazine **5**(2) (March 2007) 17–25
11. Mulliner, C., Golde, N., Seifert, J.P.: SMS of Death: From Analyzing to Attacking Mobile Phones on a Large Scale. In: Proceedings of the 20th USENIX Security Symposium. (2011)
12. Wang, P., González, M.C., Hidalgo, C.a., Barabási, A.L.: Understanding the spreading patterns of mobile phone viruses. Science (New York, N.Y.) **324**(5930) (May 2009) 1071–6
13. Hu, H., Myers, S., Colizza, V., Vespignani, A.: WiFi networks and malware epidemiology. Proceedings of the National Academy of Sciences of the United States of America **106**(5) (February 2009) 1318–23
14. Traynor, P., Butler, K., Enck, W., McDaniel, P., Borders, K.: malnets: large-scale malicious networks via compromised wireless access points. Security and Communication Networks **3**(2-3) (2010) 102–113
15. Bauer, K., Gonzales, H., McCoy, D.: Mitigating Evil Twin Attacks in 802.11. In: 2008 IEEE International Performance, Computing and Communications Conference. (December 2008) 513–516
16. Anton, B., Bullock, B., Short, J.: Best Current Practices for Wireless Internet Service Provider (WISP) Roaming. Wi-Fi Alliance. (February 2003)
17. Cranfield University: 'evil twin' hotspots are a new menace for internet users, warns cranfield university. Online: `http://www.sciencedaily.com/releases/2005/01/050120102036.htm` (Januar 2005)
18. Webkit Development Team: The webkit open source project. Online: `http://www.webkit.org/` (November 2011)
19. US-CERT: CVE-2011-0157: WebKit. Online: `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-0157` (March 2011)
20. US-CERT: CVE-2011-1290: WebKit. Online: `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-1290` (March 2011)
21. US-CERT: CVE-2011-1344: WebKit. Online: `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-1344` (March 2011)
22. US-CERT: CVE-2010-3855: CoreGraphics. Online: `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-3855` (November 2010)
23. US-CERT: CVE-2011-0226: CoreGraphics. Online: `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-0226` (July 2011)
24. Henne, B., Szongott, C., Smith, M.: Towards a mobile security & privacy simulator. In: 2011 IEEE Conference on Open Systems (ICOS2011), Langkawi, Malaysia (September 2011)

25. Savitz, E.: Windows Phone To Top iOS Market Share By 2016, IDC Says. Online: http://www.forbes.com/sites/ericsavitz/2012/06/06/windows-phone-to-top-ios-market-share-by-2016-idc-says/ (June 2012)
26. Szongott, C., Henne, B., Smith, M.: Evaluating the threat of epidemic mobile malware. In: Proceedings of the 8th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications. (2012)
27. RSA FraudAction Research Labs: New SpyEye Gains Zeus Features: Detailed Analysis of SpyEye Trojan v1.3. Online, URL modified for brevity: http://bit.ly/eAtn0B (February 2011)
28. Weintraub, S.: Google to throw the kill switch on malicious apps. Online, URL edited for brevity: http://bit.ly/eKPLQj (March 2011)

A Scripts and configurations

```
1 #!/bin/sh
2 launchctl submit -l evilTwinInstall -- /tmp/evilTwinInstall.sh
3 launchctl start evilTwinInstall
```

Listing 1.2. Post installation script of the Debian package registering the daemon script to the launch daemon

```
1  #!/bin/sh
2  name=evilTwinInstall
3  sleep 10
4  if [ -f /tmp/evilTwinInstall.success ]
5  then
6    launchctl remove evilTwinInstall
7  else
8    dpkg -i /tmp/lighttpd_1.4.18-6_iphoneos-arm.deb
9    [...]
10   dpkg -i --force-all /tmp/com.mywi4.ondemand_4.50.6_iphoneos-arm.deb
11   dpkg -i /tmp/com.mywi4_5.03.2_iphoneos-arm.deb
12
13   mv /tmp/www /var/
14   mv /tmp/com.mywi.plist /private/var/mobile/Library/Preferences/com.mywi.
         plist
15   mv /tmp/lighttpd.conf /etc/lighttpd.conf
16   mv /tmp/mypfrules.conf /etc/mypfrules.conf
17
18   launchctl submit -l webserver -- /usr/sbin/lighttpd -f /etc/lighttpd.
         conf
19   /Applications/MyWi.app/MyWiApp_ startmywi
20   pfctl -F all
21   pfctl -f /etc/mypfrules.conf
22   touch /tmp/evilTwinInstall.success
23 fi
```

Listing 1.3. Daemon script triggering the installation of all required packages