User-privacy and modern smartphones: A Siri(ous) dilemma

[1]D. DAMOPOULOS, [1]G. KAMBOURAKIS, [1]M. ANAGNOSTOPOULOS, [1]S. GRITZALIS, [2]J. H. PARK
[1]*Department of Information and Communication Systems Engineering*
University of the Aegean
GREECE
{ddamop, gkamb, managn, sgritz}@aegean.gr
[2]*Department of Computer Science and Engineering*
Seoul National University of Science and Technology
REPUBLIC OF KOREA
jhpark1@seoultech.ac.kr

Abstract

The focus of this paper is on iPhone platform security and especially on user's data privacy. We are designing and implementing a new malware that takes over the iOS mDNS protocol and exposes user's privacy information by capitalizing on the new Siri facility. The attack architecture also includes a proxy server which acts as man-in-the-middle between the device and the Apple's original Siri server.

Keywords**:** Malware; iPhone; iOS; Siri; mDNS.

1 Introduction

Over the last few years, mobile devices have experienced a rapid shift from pure telecommunication devices to small and ubiquitous computing platforms. Nowadays, such devices are equipped with enough facilities to even replace the usage of laptops. As expected, this situation draws the attention of aggressors to steal or misuse private information, or to disrupt the information flow. Typical methods to achieve such goal are gaining root permissions (known as Jailbreak [1] on iOS platform), exposing new vulnerabilities [2], and developing smart and perilous malware [3]. In fact, every new facility or service offered for modern smartphones may be susceptible to attacks. This is actually the case with the newly introduced Siri technology for the iOS platform [4]. In the following we demonstrate how an attacker can take advantage of this technology to trample on user privacy. As far as we are aware of, this is the first attack on Siri.

Siri comprises a new feature of iOS 5 the operation system of the lately introduced iPhone device 4S. It is a personal intelligent software assistant that uses a natural language interface to interact with the user in real-time and execute their voice commands. Siri communicates with a remote server via the https protocol in order to firstly translate user's voice commands to text and secondly the text commands to the corresponding actions. The exchanged data between Siri and the server can be raw audio data, property list (plist) files or other sensitive private information, such as the confidence score of each word, timestamps, the unique identifier of the device etc. So, it becomes apparent that once Siri is compromised may result in serious privacy violations. However, attacking Siri is not trivial. Specifically, as already mentioned, Siri is a proprietary software designed to communicate securely (https) with the original Siri server(s) controlled by Apple. Therefore, to fool the protocol, one has to somehow hijack the device-to-Siri_server communication in an undetectable manner. Toward this goal, we develop a malware, namely Siri Privacy Exposer (SPE), that attacks the multicast Domain Name System (mDNS) protocol [5] installed on every iOS device and then carry out a man-in-the-middle attack to take over the control of the connection. Note that Siri packet structure has been recently reversed engineered [6].

We also emphasize that in this paper we do not analyze new jailbreak methods, but use already existing ones [7] to gain root permissions on the device and infect it with our malware. It is straightforward that SPE can be integrated with other similar malware like iSAM [3], or can propagate individually by incorporating some of the existing infection methods already presented in the literature [2].

The next section briefly presents SPE architecture and provides basic information about the malware structure and the privacy information we are able to extract from the device or its user through Siri normal operation.

2 Design and Implementation

Figure 1 depicts the overall architecture of the attack scenario. Bear in mind that in a first step we must attack via SPE and compromise the iOS mDNS protocol with a view to redirect all (or selected) Internet traffic to our DNS server. The latter acts as man-in-the-middle between the iOS device and the legitimate Siri server controlled by Apple. After that, we are able to intercept user's privacy information transferred over Siri. At present, this is realized through the implementation of three custom plugins for SiriProxy [8]. SPE is written in Objective-C and compiled with Theos for iPhone ARM CPU. It is tested to run on iOS version 5 and above. Also, SPE has been built using the unofficial ways for backgrounding (daemons and dylibs), the public and private frameworks for developing iOS applications, and the MobileSubstrate framework with the substrate.h header that overrides iOS functions. This means that certain modules of SPE can be classified as rookit. The SPE core consists of a main daemon combined with a proper launch plist (activated at device boot time) and five subroutines written as Objective-C functions and dylibs. This daemon is responsible for managing all subroutines, namely NetDetector, NIUpdate, HUpdate, mDNSreloader and SirInvervine, which in turn carry out the malware tasks.

For using Siri, the device must authenticate the Siri server. This is done during the SSL handshake and the server certificate, namely guzzoni.apple.com, is pre-installed on every iPhone 4S device. Note, that the authentication is unilateral i.e., the client (device) does not authenticate itself to the service. So, to act as man-in-the-middle and hijack the https session we need to replace the original certificate with a fake one. This is accomplished by SPE. As soon as SPE infects a device, the SirIntervine subroutine executes and installs a custom SSL certification authority into iOS. This is necessary to create and sign a fake certificate for guzzoni.apple.com. After the bogus certificate is created, the HUpdate executes to populate (manipulate) the device /etc/hosts file with the IP address of our DNS server. This will redirect all Siri traffic through our DNS server. Next, NIUpdate replaces the legitimate IP address of all the known to device networks' DNS resolvers (stored in the Network Interface plist), with the one of our DNS server. At a final step, the mDNSreloader subroutine shall restart the mDNSresolver service running on the device to parse and activate the new network settings. From now on, every time the device connects to any wireless network interface, e.g., WiFi, GPRS, 3G, NetDetector is triggered so as to update the settings through the aforementioned subroutines.

Our DNS server incorporates two basic modules: (a) a typical DNS translator that redirects to any domain is configured to (in our case this is the Siri official server) and (b) the open source SiriProxy Rubi script [8] which allows us to manipulate Siri packets and create our own custom plugins to violate user privacy though the Siri technology. The server runs on a typical laptop machine which incorporates a 2.53 GHz Intel Core 2 Duo T7200 CPU and 4 GB of RAM. The OS of this machine is OS X Leopard Snow. The lightweight open source DNS Server named Dnsmasq [9] has been used as a DNS service. We also tinkered with the pre-alpha version of the SiriProxy that runs on our server to handle (i.e., decipher, encipher, modify) Siri packets.

To demonstrate the exposure of any sensitive personal information the user might exchange with the Siri facility, we conduct three real use-case scenarios. For each one, we create a custom SiriProxy plugin. According to the first scenario we successfully retrieve user's GPS location, once the user asks Siri about the weather. With only minor modifications, the plugin is able to retrieve user's location for any posed question such as "How can I get to Ocean Park?", "Where is the nearest metro station and bus stop?" etc. It is stressed that Siri obtains the geographic coordinates without directly asking the user about their location. According to the second scenario, the user sends an SMS just by speaking to Siri. During this scenario our plugin intercepts the receiver's telephone number, the SMS payload and the final outcome, i.e., whether the user finally gave their consent to send the SMS or not. For the last attack scenario we developed an even smarter plugin able not only to eavesdrop on private information but also to interact with the user and ask them custom questions. By doing so, it becomes very likely for our man-in-the-middle entity to intercept confidential information such as the user e-mail address or even the password of their e-mail account(s). Due to the fact that Siri is using artificial intelligent to interact with the user in order to accomplish a task, e.g. send out an email, the question about the password would not bear any evidence of malicious behaviour.

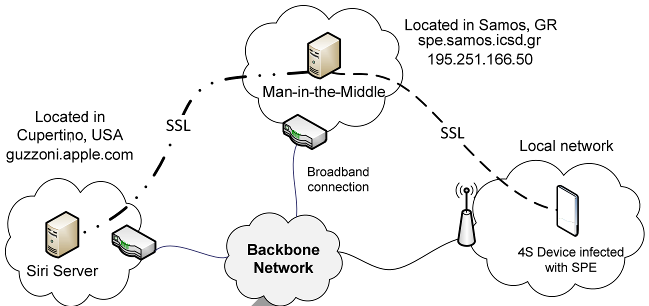

Fig. 1. General architecture and components of the attack

3 Conclusions and on-going work

SPE can be classified as DNS poisoning malware. It aims to redirect all or a subset of DNS requests to a DNS resolver which is under the control of the attacker. It is then obvious that it can severely influence the way the user experiences the Internet and expose them to serious threats. Moreover, by leveraging the Siri facility the aggressor is able to intercept sensitive user information including their geographical location, account credentials, address book, etc. Generally, such attacks stem from the fact that security and user-privacy is commonly not within the first priorities for new Operating Systems and features/services for mobile devices. In a future version of the paper we shall elaborate on SPE and explain the internal mechanics of the attack. Also, we would like to consider variations of the attack by implementing additional SiriProxy plugins.

References

[1] iOS Jailbreaking,
http://en.wikipedia.org/wiki/IOS_jailbreaking
[2] Securitygeneration, Miller discovers ios vulnerability, http://www.securitygeneration.com/security/charlie-miller-discovers-ios-code-signing-bypass-vulnerability/
[3] D. Damopoulos, G. Kambourakis, and S. Gritzalis. iSAM: An iPhone Stealth Airborne Malware, in proc. of IFIPSec 2011, vol. 354, pp. 17–28, Springer, 2011.
[4] Apple Inc., Siri, http://www.apple.com/iphone/features/siri.html
[5] Stuart Cheshire, Multicast DNS, http://www.multicastdns.org/
[6] DumasLab, Inside Siri, http://dumaslab.com/2011/11/inside-siri/
[7] MuscleNerd, iphone 4S jailbreak. https://twitter.com/#!/MuscleNerd/status/129811190066061312
[8] Plamoni, SiriProxy, https://github.com/plamoni/SiriProxy
[9] Simon Kelley, Dnsmasq, http://www.thekelleys.org.uk/dnsmasq/doc.html