Denial of Convenience Attack to Smartphones Using a Fake Wi-Fi Access Point

Erich Dondyk
College of Engineering and Computer Science
University of Central Florida
Orlando, United States of America
Email: erich@knights.ucf.edu

Cliff C. Zou
College of Engineering and Computer Science
University of Central Florida
Orlando, United States of America
Email: czou@cs.ucf.edu

***Abstract*—In this paper, we present a novel denial-of-service attack targeted at popular smartphones that are used by normal users who are not technology savvy. This type of attack, which we call a denial-of-convenience attack, prevents non-technical savvy victims from utilizing data services by exploiting the connectivity management protocol of smartphones when encountered with a Wi-Fi access point. By setting up a fake Wi-Fi access point without Internet access (using a simple device such as a laptop computer), an attacker can prompt a smartphone with enabled Wi-Fi features to automatically terminate a valid mobile broadband connection and connect to this fake Wi-Fi access point. This, as a result, prevents the targeted smartphone from having any type of Internet connection unless the victim is capable of identifying the attack and manually disabling the Wi-Fi features. We demonstrate that most popular smartphones, including iPhone and Android phones, are vulnerable to denial-of-convenience attacks. To address this attack, we propose implementing a novel Internet-access validation protocol that uses the cellular network to send a secret key phrase to an Internet validation server. Then, it attempts to retrieve this secret key phrase via the newly established Wi-Fi channel to validate the Wi-Fi access point. We have fully developed and evaluated the attacks as well as the defense prototypes that run on Android phones.**

Keywords: *denial-of-service, mobile platforms, Android, iPhone.*

I. INTRODUCTION

The smartphones are quickly taking over the mobile phone market. Currently, half of all U.S. mobile subscribers own a smartphone [1]. This rapid growth in smartphone adoption is due in grant part to the multiple services these devices can provide. However, checking email, using GPS navigation, streaming video and many other services depend on Internet connectivity. We have discovered that the majority of smartphones can be easily deprived of their Internet services, and thus, of most of their functionality through a specific form of a denial-of-service attack presented in this paper, which we call a "denial-of-convenience" (DoC) attack.

With 48% and 32.1% of the market share, Android and iPhone are by far the most popular smartphones among consumers [1]. Both of these platforms are designed to automatically switch from a mobile broadband connection (such as 3G data service) to a Wi-Fi connection whenever possible. This design allows them to take advantage of the much faster Wi-Fi Internet connection which does not utilizes the limited mobile broadband data plan of the user. However, neither of these platforms verifies whether or not the Wi-Fi access point (AP) has an Internet connection. An attacker can exploit this weakness to deny the Internet access of these smartphones.

It is very easy for an attacker to launch such a DoC attack. All it requires is setting up a Wi-Fi access point that does not have an Internet connection. This can be easily achieved via a laptop computer equipped with a cheap portable Wi-Fi adapter such as that shown in figure 2. When inside the coverage area of this fake access point, smartphones will automatically disconnect from their mobile broadband and connect to this hotspot. However, because this fake access point does not provide an Internet connection, these smartphones will be deprived of any form of Internet access.

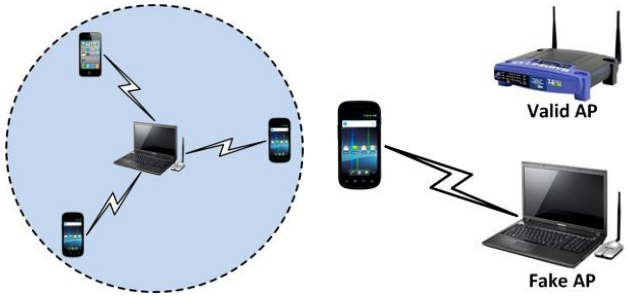


Figure 1: DoC attack scenarios. (a) The presence of a fake AP terminates the mobile broadband Internet connection of all smartphones within its coverage area and (b) when the perceived signal strength of a fake AP is stronger than that of a valid AP, a smartphone connects to the fake AP instead of the valid AP automatically and, hence, is denied of data service.

Such an attack can be resolved by a tech-savvy smartphone user. A smartphone being targeted by this attack would display an optimal network connection status. When the smartphone user notices that her phone has no Internet connection, she can manually disable the Wi-Fi function of her phone, and then her phone would automatically return back to the mobile broadband, and hence, regain Internet access. For this reason, we call the proposed attack a "denial-of-convenience" attack because it is not a hard denial-of-service to smartphone users.

However, with more than one third of all US adults currently owning a smartphone [2], we cannot expect the

majority of users to be able to diagnose this attack and successfully navigate through the solution above. Therefore, we believe this DoC attack still imposes significant threat to many smartphone users. As a result developing an automated solution to resolve this type of attack is highly desired.

To handle similar types of connectivity issues, traditional operating systems have developed several network awareness mechanisms. Microsoft's Windows, for instance, uses the Network Connectivity Status Indicator (NCSI) feature to verify the validity of an Internet connection. NCSI achieves this by sending a validation challenge to a predetermined service and comparing its response against the expected result [3]. In this paper, we develop an Android application that implements a similar network awareness mechanism. We then test its effectiveness by exposing the Android phone to a DoC attack under real conditions.

Although this type of solution is widely implemented by traditional operating systems, its effectiveness against more sophisticated DoC attacks is limited. Its weakness lies on the fact that the validation key, the value returned in the validation response, must remain constant. An attacker, therefore, can easily fool the mechanism by acquiring the static validation key and providing it to the victim when the victim's smartphone performs a network awareness test. As a result, we further develop a more robust network awareness feature capable of withstanding this type of attack. It achieves this goal by first using the cellular network to send a dynamically-generated secret key phrase to an Internet validation server, and then attempting to retrieve this secret key phrase via the newly established Wi-Fi channel to validate the Wi-Fi access point.

In short, we make the following contributions in this paper: (i) We expose a specific type of DoS attack that both iPhone and Android phones are vulnerable to. In addition, we demonstrate how it can be easily mounted on a large number of victims simultaneously. (ii) We apply a network awareness mechanism commonly used by traditional operating systems to prevent this type of attack. We implement this solution in the form of a lightweight application and test it thoroughly under real conditions. (iii) We demonstrate how this mechanism can be fooled by a more sophisticated version of the DoC attack. (iv) We present a novel solution capable of overcoming the limitations of the previously implemented network awareness mechanism which does not require user intervention. Thus, making it especially attractive because of the large number of smartphone users that cannot be expected to diagnose and solve this type of attack.

II. Related Work

Previous works on smartphone security can be organized into two subjects. The first of these subjects is smartphone malware [4], [5], [6], [7], [8], [9]. The second subject of smartphone security research considers the implementation of traditional computer exploits and defenses on mobile devices [10], [11].

Currently, there are many works available on rogue access point detection. Several enterprise Wi-Fi security systems, for example, rely on lists of authorized access points to detect when a rogue access point is introduced into an area [12] [13] [14]. ETSniffer [15], on the other hand, provides the rogue access point detection capabilities to the end user. By utilizing network metrics to detect latencies characteristics of this type of exploits, ETSniffer is able of identifying evil-twin access points with a high level of accuracy.

Our work is different from all previous works because it considers a realistic denial-of-service attack unique to smartphones. Also, we are able to successfully implement on Android a defense mechanism currently employed by tradition operating systems. Finally we propose a novel network awareness feature that relies on the mobile broadband connection of the device to provide an attack resistant authentication scheme.

III. Attacks

A. Attack I: Simple Passive Wi-Fi Access Point

The first method considered in this paper for executing a DoC attack is through a Wi-Fi access point without an Internet connection. When an Android smartphone or an iPhone enters the coverage area of a wireless access point, it is automatically assigned an identifier and loaded into the Wi-Fi stack of the smartphone. If the phone's Wi-Fi connectivity options are enabled and the access point is open or has been previously accessed, it will automatically connect to it. It will then terminate any ongoing mobile broadband connection that might have been established prior to the Wi-Fi connection. However, the smartphone does not verify, at any time during or after the connectivity process, whether the access point has a functioning Internet connection or not. Therefore, by setting up a Wi-Fi access point that is not connected to the Internet, a smartphone can be prompted to abandon its mobile broadband data connection to establish another one that does not provide any data. This, in turn, denies the user of any type of data service.

The DoC attack described above can be executed in a variety of ways. One simple approach is through a wireless router that is not connected to the Internet. This method can be implemented with little resources and technical knowledge. Another possible approach is to configure a laptop as an access point, which can be achieved by using the free network software suite aircrack-ng [16]. Then, setting up a DHCP server using dhcp3-server to automatically assign IP addresses to smartphones entering the coverage area and, thus, prompting them to connect to our fake Wi-Fi access point [17]. Because Android or iPhone gives priority to Wi-Fi access points based on their signal strength, providing a stronger signal than any adjacent valid access points would increase the chance that a smartphone connects to the fake access point. Therefore, in our prototype we utilize an external wireless adapter with an antenna so that the fake access point has a stronger signal. Figure 2 shows our prototype of the fake access point.

Furthermore, by using the same SSID as that of a valid Wi-Fi access point in the area, the attacker will be able to deprive knowledgeable users of Wi-Fi internet access. Even if the victim has an in-depth knowledge of the Wi-Fi connection

manager of the smartphone and is able to determine the connection problems are due to the fake Wi-Fi access point, he/she will not trust the valid access point in the area because it has the same SSID as the fake access point. This will prompt them to turn off the Wi-Fi features of the smartphone which, in turn, will prevent them from using the valid access point. We further demonstrate this in section A.5 of the Evaluation by executing an experiment that simulates this scenario.

Figure 2: Fake AP implementation using a Linux netbook equipped with an external ALFA network adapter costing less than $30. An attacker can easily carry this fake AP to any place to conduct the DoC attack.

Regardless of the implementation used, this type of attack can be very effective because: (i) The attacker has the ability to deny data services to a large number of victims simultaneously. (ii) The attack can be carried out in any place while in the move. (iii) From the victim's perspective, it is difficult to detect this type of attack because the device would display a nominal Wi-Fi connection status. That is, a smartphone would show that a Wi-Fi connection has been successfully established and that it is working properly. (iv) This type of attack can be executed without the use of sophisticated equipment or extensive technical expertise.

B. Attack II: Fake Validation Response

A defense against Attack I can be successfully mounted by implementing network awareness features similar to those used by traditional operating systems. These features test the Internet connectivity of an access point by sending a challenge to a validation server and comparing a validation key obtained in the response against the expected result [3]. For this to work the validation key must be known by the device before performing the validation and the validation key stored in the validation server must remain constant. However, these conditions allow an attacker to easily obtain the validation key from the validation server beforehand. Once the validation key is known, it can be used to fool the traditional network awareness procedure by responding to the probing packets with the valid answer at the time of validation. We refer to such an attack as Attack II in this paper.

In practice, there are many ways to implement Attack II. Similarly to an evil-twin access point [15], we can implement Attack II by configuring a laptop as an access point and redirecting all probing packets to a fake validation server. Because Android does not currently support ad-hoc IBSS networks [18], it is necessary to configure the computer as a full Wi-Fi access point. This can be achieved by using aircrack-ng. Then, Linux application iptables is used to redirect all probing packets to a local server that mimics the validation server [19].

Using this implementation, the network awareness protocol is able to successfully retrieve the key from the fake access point. With the correct key retrieved, the access point will be classified as valid by our testing Android phone. The connection to the fake Wi-Fi access point is maintained, the smartphone does not return to the functioning mobile broadband connection, and the smartphone user is deprived of all data services. This attack approach also has the advantage of requiring few resources. Any laptop computer with a wireless card and a UNIX/Linux operating system is sufficient to successfully execute Attack II.

C. Attack III: Selective Internet Traffic Throttling

A successful defense against Attack II could be formulated by implementing a challenge-response mechanism that relies on a dynamic key. That is, the key is different for every validation test performed. Facing this possible defense, an attacker could possibly defeat it by using a Wi-Fi access point that has Internet access, which is referred to as Attack III in this paper.

With Internet access, the fake access point could fool the dynamic-key based network awareness protocol by allowing the validation probing packets to reach the validation server while blocking all other traffic. This would allow a smartphone to successfully retrieve the dynamic validation key. As a result, the smartphone would remain connected to the fake Wi-Fi access point without any useful data service, thus successfully executing a DoC attack.

There are several ways of implementing Attack III. One possible approach is almost identical to our Attack II implementation. Essentially, it requires configuring a computer as a Wi-Fi access point, redirecting all probing packets to the real validation server, but blocking all other traffic. This can be achieved by using aircrack-ng and iptables.

Compared to Attack II, Attack III has two more requirements to be implemented. First, it requires two separate network interface cards to establish both the attacking Wi-Fi access point and a valid Internet connectivity. Second, it requires the attacking access point to have Internet access, which makes its implementation less easy than Attack II. The valid Internet access can be achieved in two possible ways. If the attacking machine is within the coverage area of a valid Wi-Fi access point (such as in a McDonald or Starbucks), it could use this valid Wi-Fi to access the Internet; if there is no valid Wi-Fi, the attacker could use a mobile broadband modem to connect to the Internet.

IV. DEFENSES

A. Defense against Attack I: Static Identifier Validation

In order to counteract Attack I, we implement a network awareness protocol on Android very similar to the NCSI feature currently used by Microsoft's Windows [3]. This protocol, implemented as an Android app, which we call Wi-Fi Authenticator, automatically verifies whether or not the

currently connected Wi-Fi access point has a functioning Internet connection without the need for any user intervention.

To achieve this, Wi-Fi Authenticator relies on the following two-step process: (i) Every time a Wi-Fi connection is established with an access point, Wi-Fi Authenticator sends a challenge to a validation server. If a response is not obtained within some time period, the access point is considered invalid. On the other hand, if a response is received, Wi-Fi Authenticator proceeds to step 2 of the validation process. (ii) Wi-Fi Authenticator retrieves the validation key from the validation response and compares it with the validation key stored in the smartphone. If the two validation keys match, the access point is considered valid. Otherwise, it is considered invalid. Step 2 prevents an attacker from easily fooling the authentication protocol by sending an arbitrary response to any challenge. In this approach, any website could be used as validation server. For example, Google.com could be the validation server and the word "google" the validation key.

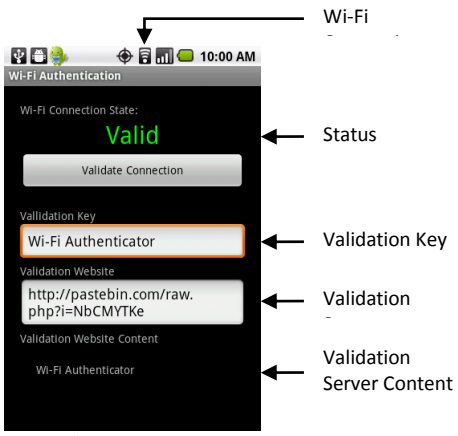


Figure 3: Result of a Wi-Fi Authenticator validation test in which the AP was determined to be valid.

If the Wi-Fi access point is considered invalid in either step, Wi-Fi Authenticator terminates and disables the connection. This prompts the Android smartphone to transit back to its mobile broadband data connection, and hence, returning Internet data services to the user. Also, it continues to enable the Wi-Fi capabilities of the device, allowing it to connect to other Wi-Fi access points that might become available in the future.

B. Defense against Attack II: Dual Channel Validation

As demonstrated by Attack II, a more sophisticated attacker with greater technical knowledge could overcome Defense I introduced above. This is primarily due to the fact that the validation key used in Defense I is constant (the same problem exists in the network awareness protocol used in Windows). In order address this weakness, we propose a dual-channel network awareness protocol in which the validation key changes every time a validation test is performed. We achieve this by relying on the unique mobile broadband data channel of smartphones, which cannot be easily hijacked or blocked by an attacker. An attacker, therefore, is unable to fool the protocol by supplying the expected response because it is unknown to her.

This approach relies on the following five-step process to validate a Wi-Fi access point: (i) After encountering an accessible Wi-Fi access point, the smartphone generates a random key and sends it along with its MAC address to the validation server through the cellular network. Depending on the user's billing agreements, this data can be sent as an SMS or a TCP packet. (ii) The validation server stores the random key in a table using the MAC address of the smartphone as the index. (iii) After transitioning from the mobile broadband to the Wi-Fi connection, the smartphone sends a challenge to the validation server with its MAC address. (iv) The validation server responds with the key corresponding to the smartphone's MAC address. (v) The key obtained from the validation response is compared against that generated earlier by the smartphone. The Wi-Fi connection is considered valid if these two keys match. Otherwise, it is considered invalid.

Similarly to Defense I, this validation test is performed automatically without the need for any user intervention. Also, if a Wi-Fi access point is considered invalid, the connection is terminated and disabled. This allows the device to regain Internet services by reconnecting to the mobile broadband while maintaining its Wi-Fi capabilities enabled.

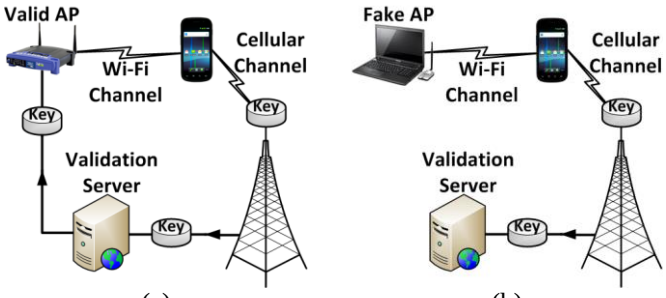


Figure 4: Dual-channel validation protocol: (a) the real AP is able to provide the secret key to the smartphone for validation and (b) the fake AP cannot produce the secret key because it is not connected to the Internet, and hence, the attack is detected.

C. Defense against Attack III: Network Performance Monitoring

Given the case of Attack III in which the attacker has Internet access, Defense II can be compromised. The effectiveness of Defense II lies on the fact that the key is not known prior to the validation test. However, if the attacker has an Internet connection, it could execute a successful DoC attack by allowing the challenge to reach the validation server but blocking or throttling all other traffic.

This weakness can be eliminated by expanding Defense II into considering network measurement. If traffic blocking or throttling occurs, the network awareness protocol would measure the network performance and detect that it is below a predetermined threshold (e.g., too high packet loss ratio). Then, it would regain data services by prompting the smartphone to transition back to the mobile broadband by disabling the Wi-Fi access point.

There are many different metrics that could be used to realize this defense. Packet drop, instant throughput, and average throughput are among the many approaches that could be taken. In [20], [21], and [22] some of these metrics are explained and their advantages and disadvantages explored.

V. Evaluation

In this section we evaluate our real implementations of the proposed Attack I, Defense I, Attack II, and Defense II. Attack III is exactly the same as Attack II but with the addition of an Internet connection to our testing laptop. Defense III will not be further explored in this paper because network measurements have been well studied before. The source code of the attack scripts, Wi-Fi Authenticator app, and the validation server used throughout the evaluation can be found at www.cs.ucf.edu/~czou/denialofconvenience/.

A. Attack I: Simple Passive Access Point

1) *Vulnerability to Android:* To examine the vulnerability of Android to a DoC attack, we setup a typical household wireless router without connecting it to the Internet. We enter the coverage area of this Wi-Fi access point with an Android smartphone in sleep mode. Then, we awake the phone and perform a Google query using the default browser of the device. We perform this test on three different phones running Android 2.1, Android 2.2, and Android 2.3. In all cases, Android automatically connects to the fake Wi-Fi access point immediately after coming out of sleep mode and, hence, is unable to perform the query. After approximately 15 seconds, it displays that the web page is not available. As shown in figure 6, Android displays a strong Wi-Fi signal throughout this process.

2) *Vulnerability to iPhone:* To evaluate the vulnerability of iPhones to a DoC attack, we execute the test previously described on an iPhone 4S. By default, the iPhone asks the user before connecting to a Wi-Fi network encountered for the first time. However, after the user selects our Wi-Fi access point, the iPhone behaves exactly like Android. It automatically connects to the fake Wi-Fi access point immediately after coming out of sleep mode and is unable to perform the Google query. However, the iPhone never displays a message informing the user that the query could not be completed. Instead, it continues attempting top perform the query until the screen timeouts and the phone goes back to sleep.

3) *Interruption of Existing Mobile Broadband Connection:* To examine the effects of a DoC attack on smartphones already connected to the mobile broadband, we perform the following test on Android 2.1, 2.2, 2.3, and iPhone 4S, respectively. First, we begin downloading a large file through the mobile broadband connection. Then, we introduce a fake Wi-Fi access point in the middle of the download. When the fake Wi-Fi access point becomes available, Android and iPhone immediately disconnect from the mobile broadband and connect to the fake access point. As a result, they are unable to complete the download.

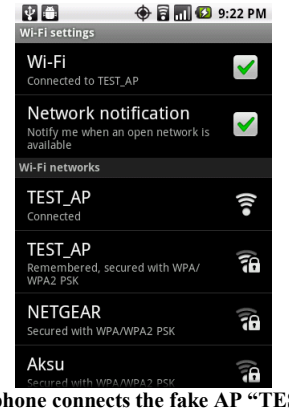
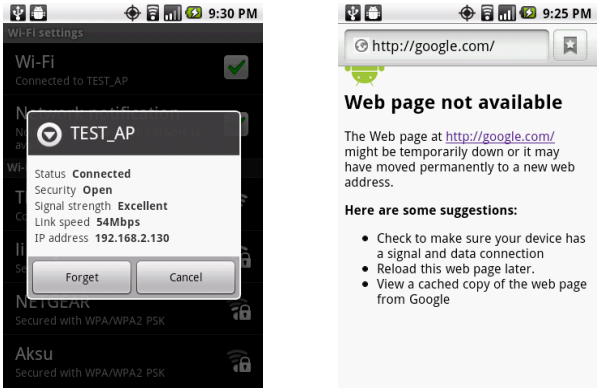


Figure 5: The smartphone connects the fake AP "TEST_AP" even though there is a valid AP with the same SSID in the area.

4) *Defeating Existing Valid Wi-Fi Access Points:* If the fake access point is introduced while there is an ongoing connection with another valid Wi-Fi access point, our testing shows that the existing connection will not be interrupted. However, if the smartphone is put to sleep by the user or after a period of inactivity (Android and iPhone can enter sleep mode in as little as 30 seconds if not touched), it reconsiders all Wi-Fi access points in the area when awaken. As a result, the smartphone connects to the fake access point when it perceives that the fake access point has the strongest signal.

In order to simulate a realistic scenario, we execute Attack I in a location where there are several valid Wi-Fi access points available. We use aircrack-ng to configure a Linux laptop computer as our fake Wi-Fi access point and dhcp3-server to authenticate devices entering its coverage area. Because Android gives higher priority to Wi-Fi access points with stronger signals, we utilize the AWUS036H ALFA network adapter shown in figure 2 to increase the success probability of the attack. With an output power of 30dBm [23], this adapter is able to overcome other typical wireless routers that have an average output power of 20 dBm [24].



(a) (b)

Figure 6: The result of Attack I on an Android phone: (a) the connection status of the fake AP and (b) the smartphone does not have a working Internet connection because of its Wi-Fi connection with the fake AP.

To execute our test, we first set up our fake Wi-Fi access point in a room such that it exhibits a stronger signal than the

other valid Wi-Fi access points covering the area. We enter the room with an Android 2.1, 2.2, 2.3, and an iPhone 4S respectively and perform an arbitrary Google query using the default browser on the smartphone. As shown in figure 6, the smartphone always connects to the fake access point because it provides a stronger signal than the other valid access points. However, at the time of a second Google query, no information is returned nor a notification shown. Thus, a successful DoC attack is achieved.

5) *Fooling Knowledgeable Users by Using a Valid SSID:* Even if the user is sufficiently technical to turn off the Wi-Fi features of the smartphone, a DoC will have a negative effect. We demonstrate this by implementing a fake access point that has the same SSID as a valid access point in the area. By utilizing the AWUS036H ALFA network adapter, the fake access point is capable of producing a stronger signal than the valid access point. As figure 5 illustrates, the smartphone connect to the fake access point because the stronger signal increases its priority in the Wi-Fi stack.

Because the fake AP has the same SSID as the valid AP, even if the user is capable of determining the connectivity issues of the smartphone are due to the Wi-Fi connection, he/she will attribute the problem to the valid access point and, as a result, disable the Wi-Fi features of the device to terminate the connection. However, this will keep them from taking advantage of the valid Wi-Fi access point in the area.

B. Defense I: Static Identifier Validation Protocol

We implement our proposed Defense I network awareness feature as an Android app which we call Wi-Fi Authenticator. Wi-Fi Authenticator tests the Internet connectivity of an access point by accessing a website, retrieving its content, and comparing it against a key phrase. This validation scheme is performed automatically every time a Wi-Fi connection is established. If Wi-Fi Authenticator is unable to access the website or if the content retrieved does not contains the validation key, the access point is considered invalid and disabled. For these tests, we use the Google homepage (74.125.227.1) as the validation website and the word "google" as the validation key. Because the html of Google's homepage contains several occurrences of the word "google", we are able to use this configuration for proof of concept. However, any website with an expected phrase could be used to achieve the same results.

To evaluate the performance of Defense I, we execute an experiment very similar to that of Attack I. We expose an Android 2.1, 2.2, and 2.3 smartphone respectively with Wi-Fi Authenticator to the fake AP. In all cases, Wi-Fi Authenticator is able to determine the Wi-Fi access point is invalid, disconnect from it, and reconnect to the mobile broadband. Figure 7 shows the detection of the fake access point by the Wi-Fi Authenticator app installed on the Android phones.

Wi-Fi Authenticator is able to determine whether or not a Wi-Fi access point is valid almost immediately. However, the time it takes for Wi-Fi Authenticator to detect a fake access point varies. Figure 8 shows these variations in validation times for fake Wi-Fi access points.

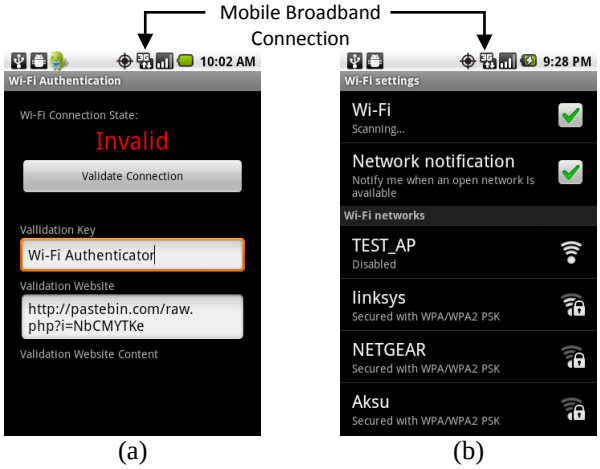


Figure 7: Detection of the fake AP "TEST_AP" using Wi-Fi Authenticator: (a) result of Wi-Fi Authenticator validation test and (b) Wi-Fi stack of the smartphone showing that the fake AP has been disabled by Wi-Fi Authenticator.

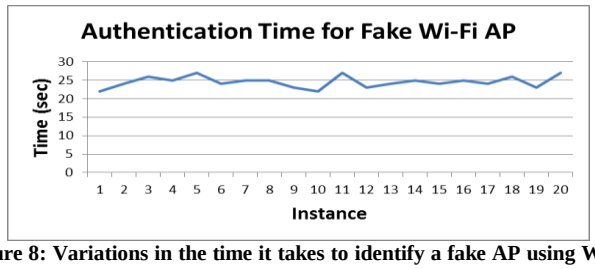


Figure 8: Variations in the time it takes to identify a fake AP using Wi-Fi Authenticator.

C. Attack II: Fake Validation Response

To implement Attack II, we first create a fake validation server in the attacker's laptop computer by setting up an Apache HTTP Server [25]. Because Wi-Fi Authenticator uses "google" as the validation key of our validation scheme, the Apache server is configured to respond with a webpage containing the word "google". Just like in our Attack I implementation, we also configure the attacker's computer as a Wi-Fi access point using aircrack-ng and a DHCP server using dhcp3-server. Finally, we use iptables to redirect all traffic sent to Google's homepage (74.125.227.1) to the IP address of the fake Wi-Fi access point network interface.

To measure the performance of Attack II, we enter the coverage area of our attacker's laptop computer with an Android 2.1, 2.2 and 2.3 smartphone respectively that have Wi-Fi Authenticator installed. This test is repeated 20 times erasing the Wi-Fi stack of the smartphone between each test. In all cases, Wi-Fi Authenticator is unable to determine that the fake Wi-Fi access point is invalid. Consequently, the Wi-Fi connection is preserved and the smartphone is denied Internet connectivity.

D. Defense II: Dynamic Identifier Validation Protocol

In order to implement Defense II, we expand the Wi-Fi Authenticator app used for Defense I. Immediately after the smartphone begins the authentication process with a Wi-Fi access point, Wi-Fi Authenticator generates a six-digit random number. This number is sent through the mobile broadband

connection to our validation server. Our validation server, implemented in the form of a desktop computer running a HTTP server developed in Java, receives and stores the random number in a table for two minutes. Finally, just like in Defense I, Wi-Fi Authenticator tests the connectivity of the access point by accessing the server, retrieving its content, and comparing it against the random number generated earlier.

We test Defense II with the same method used before. First, we setup the Attack II fake access point. Then, we expose an Android 2.1, 2.2 and 2.3 smartphone respectively that has our enhanced Wi-Fi Authenticator app installed to the fake access point. In all cases, Wi-Fi Authenticator is able to determine that the Wi-Fi access point is invalid, disconnect from it, and reconnect to the mobile broadband.

VI. Conclusion

In this paper, we have considered a new form of denial-of-service attack targeted at popular smartphone operating systems. We present three possible approaches for executing this attack along with three defenses capable of counteracting them. We demonstrate, through real implementation and testing, that such attacks are successful at achieving their purpose. Also, we demonstrate how each proposed defense is capable of counteracting the different implementations of the attacks. Our network awareness implementation is able to automatically validate a Wi-Fi access points in less than a minute from the background, imposing no operation burden on smartphone users.

References

[1] Palis, C. Smartphone Market Share: Devices Make Up Almost Half Of All Phones, With 2 Players Gunning For Top Spot. http://www.huffingtonpost.com/2012/03/29/smartphone-market-share_n_1388368.html, 2012.

[2] Smith, A. Smartphone Adoption and Usage. 2011 http://www.pewinternet.org/Reports/2011/Smartphones.aspx

[3] Appendix K: Network Connectivity Status Indicator and Resulting Internet Communication in Windows Vista. http://technet.microsoft.com/en-us/library/cc766017%28WS.10%29.aspx.

[4] Backes, M., Gerling, S., and Styp-Rekowsky, P. A Local Cross-Site Scripting Attack against Android Phones. https://www.infsec.cs.uni-saarland.de/projects/android-vuln/android_xss.pdf>, 2011.

[5] Sastry, B. V. S. S. R. S., and Akshitha, K. Authorizing Stockpile Attacks on Android. International Journal of Mathematical Archive, 2.11:2475-2479, 2011.

[6] Vidas, T., Votipka, D., and Christin, N. All Your Droid Are Belong To Us: A Survey of Current Android Attacks. In Proceeding of the 5th USENIX Workshop on Offensive Technology (WOOT '11), San Francisco, CA, August 8-12, 2011.

[7] Schmidt, A. D., Schmidt, H. G., Batyuk, L., Clausen, J. H., Camtepe, S. A., and Albayrak, S. Smartphone Malware Evolution Revisited: Android Next Target? In Proceeding of the 4th Annual Malicious and Unwanted Software (MALWARE '09), Montréal, Canada, October 13-14, 2009, 1-7, 2009.

[8] Porter Felt, A., Chin, E., Hanna, S., Song, D., and Wagner, D. Android Permissions Demystified. In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS '11), Chicago, IL, October 17-21, 2011, ACM, New York, NY, 627-638, 2011.

[9] Nauman, M., Khan, S., and Zhang, X. Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10), Beijing, China, April 13-16, 2010, ACM, New York, NY, 328-332, 2010.

[10] Kumar, N., and Ul Haq, M. Penetration Testing of Android-based Smartphones. Master's Thesis. Chalmers University of Technology, Gothenburg, Sweden, 2011.

[11] Portokalidis, G., Homburg, P., Anagnostakis, K., and Bos, H. Paranoid Android: Versatile Protection For Smartphones. In Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC '10), Austin, TX, December 6-10, 2010, ACM, New York, NY, 347-356, 2010.

[12] AirMagnet WiFi Analyzer. http://www.flukenetworks.com/enterprise-network/wireless-network/AirMagnet-WiFi-Analyzer.

[13] AirWave.http://www.arubanetworks.com/products/management-security-software-2/airwave.

[14] WiSentry – Wireless Access Point Detetion System. http://wimetrics.com/Products/WAPD.htm.

[15] Song, Y., Yang, C. and Gu, G. Who Is Peeping at Your Passwords at Starbucks? – To Catch an Evil Twin Access Point. In Proceeding of the 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '10), Chicago, IL, June 28-July 1, 2010, 323-332, 2010.

[16] Aircrack-ng. http://www.aircrack-ng.org/index.html.

[17] ISC DHCP Server. http://www.isc.org/software/dhcp.

[18] MIC_888. Android Ad-hoc Wireless Network Support. http://www.xda-developers.com/android/android-ad-hoc-wireless-network-support/, 2010.

[19] iptables. http://www.netfilter.org/projects/iptables/index.html.

[20] Boschi, E., D'Antonio, S., and Schmoll, C. Network Performance Metrics and Measurement Methods in IP Networks. Technical Report. European Telecommunications Standards Institute, 2008.

[21] Bandwidth, Packets Per Second, and Other Network Performance Metrics. http://www.cisco.com/web/about/security/intelligence/network_performance_metrics.html.

[22] Cox, G. W. Network Metrics. http://www.cs.uah.edu/~gcox/570/570lec02-backgroundB-F07.pdf, 2007.

[23] ALFA Network AWUS036H. http://www.alfa.com.tw/in/front/bin/ptdetail.phtml?Part=AWUS036H&Category=0.

[24] Oney, M. Wireless Router Capacity. http://www.wirelessforums.org/wireless-networking-discussion/wireless-router-capacity-43644.html.

[25] Apache HTTP Server. http://httpd.apache.org/.