Using Deep Packet Inspection to Detect Mobile Application Privacy Threats

Gordon Brown
University of Colorado, Corlorado Springs
Colorado Springs, CO 80918
Email: gbrown@uccs.edu

Kristen R. Walcott
University of Colorado, Corlorado Springs
Colorado Springs, CO 80918
Email: kwalcott@uccs.edu

***Abstract*—Modern mobile and embedded devices hold increasing amounts of sensitive data, with little visibility into where that data is sent. Existing solutions tend to be platform-specific, targetting only Android or iOS.**

We present a technique and a develop a tool that can detect exfiltration of private data in a platform-independent way by utilizing deep packet inspection. We examine the techniques and patterns common to sensitive personal data attacks and evaluate our tool's effectiveness in detecting the exfiltration of sensitive data on known mobile malware. We learn that the majority of the known-malicious applications in our study were unable to exfiltrate contact information. Only one known-malicious application under test was able to connect to a remote server and send contact information. One known-malicious application under test was able to connect to a remote server and send contact information, which raises awareness in the software assurance community.

Keywords: Mobile Privacy, Security, Information Assurance, Malicious Applications

I. Introduction

As mobile and connected device usage continues to grow, the type and amount of data collected, either by actively malicious applications or by "free" advertising-supported applications is a growing concern in the minds of users. Users are generally more concerned about some types of data than others, but most are especially uncomfortable with sharing data such as text messages, contact lists, and calendar information. Having this information stolen can lead to serious consequences. For example, if a user's text messages are stolen and that user engages in mobile banking, the stolen text messages could be used to obtain access to that user's bank account. Stolen contact information could lead to spear phishing attacks against friends or coworkers. Fueled by this concern, there is a growing body of research into ways to detect applications which transmit user data to undesirable destinations, but there are still severe limitations on the techniques available.

There is a wealth of research into analyzing mobile applications, both in the realms of static analysis such as ScanDal [11] and dynamic on-device analysis such as TaintDroid [8]. However, these approaches are specific to each platform - a separate framework is required for the analysis of applications on other platforms. An example is PiOS [7] for Apple iOS applications. This leads to a lack of analysis tools for less popular and more recent platforms, such as Windows Phone or Blackberry, or devices which are completely proprietary, such as Internet of Things (hereafter "IoT") devices.

The lack of tools applicable to IoT devices is particularly concerning - an ever-increasing number of household object are available in Internet-connected form, from thermostats [22] to cookware [13]. Because the manufacturers of these devices generally do not distribute copies of their software, even in binary form, they are particularly difficult to analyze for security or privacy problems

Analyzing the network traffic from mobile applications or connected devices can reveal with certainty what data is being transmitted and where it is being sent to in a platform-independent way. A tool designed to automatically inspect network traffic can warn the user when potentially private data is sent to an unknown destination. Many of the data categories users are most concerned about sharing (again, per Ferreria, et al.), including "Messages & Calls", "Contacts", and "Browser" are most likely to include somewhat structured data such as telephone numbers, email addresses, and URLs in close proximity to timestamps, durations, text blobs, or audio files. This type of data is most likely to be easily recognized by an automatic traffic inspection tool.

Research into using network analysis techniques to detect privacy leaks has been very limited to date, particularly with respect to end user privacy protection. Network-based analysis allows platform independent data protection in a way that is sorely needed in this age of proprietary platforms, both in the mobile and embedded realms.

In this paper, our main contributions are:

- A technique for detecting exfiltration of private data over a local wireless network using deep packet inspection.
- A tool which implements the above technique.
- An analysis of several mobile applications that demonstrate the effectiveness of the deep packet inspection technique.

II. Identification of Private Data

There are two main components to identifying exfiltration of private data: Acquiring the transmissions that may contain private data and identifying private data in those transmissions.

A. *Network Traffic Interception*

Automatic analysis of data transmitted over a network, also known as Deep Packet Inspection, is used in existing systems to detect exfiltration of sensitive data from high-security networks [21], such as government networks. Fundamentally, inspecting traffic to detect transmission of private user data is not very different, however, the specific techniques involved do vary.

In order to analyze network traffic, the traffic must be observed. For reasons of efficiency and privacy, networks typically do not transmit, or route traffic to devices which are not necessary to get the traffic to its intended destination. There are several ways to force traffic to be directed through a specific device so that it may be analyzed. Some of these include Network Tapping, ARP Spoofing, or the use of proxy servers.

Network tap
: Network tapping involves inserting a device directly into the networking flow, similar to a phone tap. This approach requires specialized and expensive hardware, and is most often used by systems which passively monitor network traffic, such as NETSCOUT's TruView network performance analysis systems.

ARP Spoofing
: ARP, or Address Resolution Protocol, is used by computer networks to identify which devices on a local network traffic should be sent to, given an IP address. ARP is susceptible to an attack which allows a single device to claim all IP addresses, forcing all traffic to flow through the attacking device before being forwarded on to other devices. This attack can be detected by security systems [1] on high-security networks, but has been used with success in traffic inspection devices targeted at home networks, such as Disney's parental control system Circle [4].

Proxy server
: Many devices include support for configuring a proxy server or Virtual Private Network (VPN), which automatically forwards all traffic through that server. This is often done to allow access into or out of a highly secured network. However, this requires configuration on each device to be monitored, and many IoT devices do not have the ability to use proxy servers.

All of these techniques could be termed "man-in-the-middle attacks," as they seek to allow a party which is not the intended recipient of network traffic to inspect or modify it en route. However, each of these techniques are also used for non-malicious purposes, such as network troubleshooting, parental control [4], or advertisement blocking [5], because these things require some level of analysis of network traffic without being the intended recipient of that traffic.

B. Identifying Private Data

The categories users are least comfortable sharing with third parties are "Messages & Calls" and "Contacts" [9]. Both of these categories include the contact information of others: There would be relatively little value in stealing messages or call records without knowing who they were sent to or from, and "Contacts" is composed entirely of this data.

The most common contact information stored on mobile devices are telephone numbers and email addresses. Both of these types of information are highly structured - telephone numbers taking the form of 7 to 11 numbers, depending on the inclusion of area and country code, with some optional delimiters (i.e. the parentheses which typically surround the area code, or the dash separating the first three and last four digits). Email addresses take the form of "{individual}@{domain}.{tld}", where "{individual}", "{domain}", and "{tld}" take the form of one or more typically alphanumeric characters, but may include periods or Unicode characters. Such highly structured data is recognizable efficiently through the use of relatively simple regular expressions. Regular expressions can be converted to deterministic automata [2], which can be run very quickly.

More complex data, such as calendar events or documents (which users are also mostly uncomfortable with sharing, per Ferreira, et. al.) are typically more complex and require more complicated heuristic techniques.

III. Implementation

Given the challenges, we design and implement a tool which intercepts and inspects network traffic for the purpose of detecting the exfiltration of private data via a local Internet-connected network. We investigate the effectiveness of this technique using the tool described.

A. Traffic Capture

Data collection is accomplished through the use of the ARP spoofing technique described above, taking inspiration from the man-in-the-middle attack suite Ettercap [15]. To accomplish this, forged ARP packets are broadcast by the program, claiming to own the addresses of both the device under test and the Internet gateway. Sending ARP packets without requests is allowed by the ARP specification [17], and these unrequested ARP packets are known as "ARP Announcements" or informally as "Gratuitous ARPs". These announcements are sent every 1.5 seconds, in our implementation, as well as whenever other ARP packets are detected on the network. This ensures the program misses little or no traffic from the device under test.

After being captured for analysis, the traffic is forwarded to its intended destination using the built-in Linux IP Forwarding feature, the same way Linux-based routers operate. This is possible as the operating system running our program does not receive the forged ARP packets, and thus uses the unaltered IP address to Ethernet address pairings.

In addition to the ARP spoofing mode of capturing traffic to analyze, the tool can also be run on saved packet captures produced by other tools, such as Wireshark or Ettercap.

B. Traffic Analysis

We postulate HTTP traffic is the most likely to be used to communicate private information to a hostile server, due to its

Malware Name	**Contact Exfiltration (Manually Detected)**	**Contact Exfiltration (Automatically Detected)**
Beita	No[1]	No[1]
DougaLeaker	No[1]	No[1]
FakeDaum	Yes	Yes
Godwon	No[1]	No[1]
Godwon2	No[1]	No[1]
Loozfon	No[1]	No[1]
Scipiex	No[1]	No[1]
Simhosy	No[1]	No[1]

TABLE I
MALWARE TESTED

Application Name	**Contact Exfiltration (Manually Detected)**	**Contact Exfiltration (Automatically Detected)**
Popular Music Funny Ringtone	No	No
Sexy Girlfriend Fake Call	No	No
Uber	No	No
Facebook	No	No

TABLE II
APPLICATIONS TESTED

```
\+?\d?([\s-.\/]?)((\(\d{3}\)?)|(\d{3}))([\s-.
\/]?)(\d{3})([\s-.\/]?)(\d{4})
```

Fig. 1: A Regular Expression for Finding Telephone Numbers

```
Suspicious data sent to 103.30.7.178: +685-555-1234
Suspicious data sent to 103.30.7.178: +555-555-3348
Suspicious data sent to 103.30.7.178: +723-555-2424
```

Fig. 2: Sample Output

ease of use in mobile applications and servers. Thus, we focus on the analysis of HTTP traffic.

Collected traffic is divided into "flows" based on source and destination IP address, as well as source and destination TCP port numbers. This allows us to reassemble HTTP transactions, which are then searched based on patterns likely to occur when transmitting private data. For example, a regular expression such as the one in Figure 1 can recognize telephone numbers. If a telephone number is detected in outgoing traffic, then the telephone number can be extracted, marked as suspicious data, and presented to the user for inspection. Email addresses can be similarly extracted and reported.

Additional pieces of data detected include timestamps in several human-readable formats (such as those specified in ISO8601 [10]), which are suspicious when in proximity to telephone numbers or email addresses, as this may indicate transmission of text messages or emails. GPS coordinates are also recognized through similar means, and are likely to represent location information.

The marked transmissions can then be presented to the user for closer examination, along with the destination of the transmission. This is done through printing the information to the screen, as seen in Figure 2, as well as exporting the packets in the flow to a standard packet capture ("pcap") file, for analysis in common packet analysis tools such as Wireshark. This allows the user to determine if private information has been leaked, and if so, which information it was, as well as where it was going.

[1]As noted in the text, this malware was unable to send private data because the server it attempts to send to has been taken offline.

IV. EVALUATION

We show that inspection of network traffic is an effective means of detecting personal data exfiltration by malicious mobile applications and notifying the user of what information was stolen and where it was sent.

Evaluation was performed by installing and running known-malicious mobile applications collected from Contagio Mobile [16], a repository of Android and iOS malware and running these applications, while connected to a private, password-protected wireless network, which the network analysis tool was also running on. Several well-reviewed applications are run as well, in order to seek false positives. Futher, multiple lesser-known applications from the Google Play Store were installed and run to attempt to find data exfiltration in live applications. Finally, some traffic constructed to show signs of data exfiltration (using HTTP POST to send a list of phone numbers and email addresses) and traffic recorded from filling out a web form were analyzed. All traffic analyzed was saved, unmodified, for manual inspection in order to confirm results.

A. Evaluation Environment and Setup

Running on a private wireless network with no other devices connected ensures that there is no accidental violation of privacy by collecting the traffic of individuals not part of the experiment - all traffic collected is explicitly either sent from or to ether the device running the application under test, or the device running the network analysis tool, both of which are owned by the author who has, of course, given his full consent.

Additionally, using a private network with only the network analysis tool and the device under test connected minimizes the threat of network traffic not associated with the device under test producing false positives or otherwise interfering with results.

The device used to test the various applications is a Samsung Galaxy Nexus running Android 4.3. An Android device was selected for the test because Android devices are commonly used, contain a significant amount of sensitive data, and often run third-party applications. However, as noted in the introduction, the results should hold across all platforms.

The Android device was prepared for testing by entering a number of false contact list entries, text messages, and emails that are easily recognizable so as to be . The device was run without a connection to a mobile network in order to avoid text messages or calls being send to "premium" numbers which charge money, a common tactic in mobile malware. This also forces all network traffic over the local WiFi network, which allows the capture of all network traffic.

B. Traffic Capture and Analysis

Each application, malicious and not, were installed, tested, and uninstalled before installing the next application in order to prevent previously installed applications from tainting the result of each application. Each malicious application is opened and used for a short period for its normal use (which varies by application) while traffic is captured.

Traffic capture for each application is done twice, once using the tool we present and once using an application available from the Google Play Store, "Packet Capture" [20], which uses the "Proxy Server" approach described above, with the proxy server running locally on the Android device in question. This application uses Android's built-in VPN facility, and ensures that all traffic will be compared.

Finally, we constructed some network traffic to simulate entering an application sending private data over the network with a simple HTTP POST, as well as submitting private data into a web form.

C. Results

The traffic captured from each application is inspected by hand, using the open-source tool Wireshark [3] to determine if there is any data being exfiltrated. This is compared to the results produced by our tool, which prints any suspicious data to the screen.

Several malicious applications known to exfiltrate contact list information acquired from Contagio were installed sequentially on the device under test, being sure to thoroughly remove each application before installing the next.

However, the majority of the known-malicious applications were unable to actually exfiltrate contact information, as the remote server they attempt to connect to has been taken offline. Only one known-malicious application, FakeDaum, was able to connect to a remote server and send contact information. This data exfiltration was detected and reported as intended, and the exact output of the program can be seen in Figure2. The phone numbers shown in that figure are from contact list entries on the Android device running the malware. The results of running this malware can be seen in Table I.

Additionally, several applications from the Google Play Store were tested. Both well-reviewed applications and applications with few reviews and permissions that would allow the exfiltration of contact information were tested. However, none of the applications tested attempted to exfiltrate any contact information that was visible through either manual inspection or through automatic detection. The results of the analysis of the traffic from these applications can be seen in Table II.

Our network analysis tool correctly identified all of the "artificial" cases described above. The first case of constructed traffic shows that detection is not unique to the "FakeDaum" application, and the latter case of a web form demonstrates a known false positive.

D. Discussion of Results

Our tool correctly identified all traffic which was either constructed to contain the transmission of sensitive private data or contained the transmission of sensitive data which could be identified manually. This shows the effectiveness of detecting exfiltration of private data through interception and analysis of network traffic, with a minimum of false positives.

Despite the issues with the known-malicious applications being unable to connect to their associated remote servers, we can still ascertain some useful information. From the connection requests, all of the above applications were attempting to connect to TCP port 80, which indicates that they are using unencrypted HTTP connections, rather than encrypted HTTPS connections which would likely use TCP port 443. This implies that analysis of traffic to that server, if it had been present, would have revealed any sensitive data being transmitted, given the success with constructed data transmission, as well as with the traffic from FakeDaum.

V. Related Work

Existing systems for data exfiltration detection tends to focus on detecting transmission of a subset of a known corpus of documents [14]. This most useful in corporate or governmental environments, where there exist many sensitive documents, such as confidential business plans or classified information. A typical end user may have some such documents, such as past tax returns, but is also concerned about data which is typically much more easily accessible by a malicious mobile application, such as contact information. Such information does not fit the mold required for existing data exfiltration detection techniques, but is likely to fit more general signatures. For example, most telephone numbers will be seven, ten, or eleven digits depending on the inclusion of area and country code. Our approach, using these general signatures, allows for increased flexibility over traditional data exfiltration detection techniques.

Additionally, existing systems for detecting data exfiltration assume a highly competent adversary [21], which effectively utilizes encryption and/or data obfuscation techniques to hide data theft and avoid detection. This is appropriate for systems used to protect high-value information such as government secrets, but given that malicious mobile applications are stealing relatively low-value data and thus are likely to aim to acquire as much as possible with the least effort, in conjunction with the low rate of cryptography use (and especially secure cryptography use) in mobile applications overall [6], these assumptions are unlikely to be valid for this threat model. By designing for assumptions appropriate to home users, we can vastly reduce the amount of processing power required as well as increase the accuracy of exfiltration detection.

There has been some very limited research that is very similar to the proposed solution - AntMonitor [12] in particular is very similar. AntMonitor uses a VPN-based method of traffic collection, which has the advantage of working whether on wireless Internet or on cellular networks. However, the prevents it from working with Internet of Things devices. Additionally, it relies on information gathered from an application running on the device under test to determine what strings to look for - our system monitors traffic based on patterns, and thus is independent from the device under test.

Similar to AntMonitor, Meddle [18] also uses a proxy/VPN approach to traffic capture, and searches for some of the same information that our analysis tool does, that is, PII. However, our analyzer notifies the user in real time of transmission of sensitive information, and as discussed, our tool's use of ARP poisoning rather than a proxy-based approach allows usage with a wider range of devices.

The open-source tool "ngrep" [19] (network grep) allows the usage of regular expressions on network traffic, and has existed for some time. This is similar to the technique our tool uses to detect suspicious transmissions, but we also include a component to intercept network traffic through the use of ARP poisoning.

VI. Threats to Validity

The primary threat to this approach is the use of encryption by malicious devices to obfuscate the data they are exfiltrating. The most obvious one is the use of Transport Layer Security (TLS) or Socket Security Layer (SSL), collectively known informally as "HTTPS". However, the use of these technologies is not ideal for the developer of a malicious application - HTTP clients typically check the public keys of SSL/TLS against centralized repositories known as Certificate Authorities to ensure the public key transmitted by a server is one known to be associated with the owner of the server. If it is not, this signals a likely man-in-the-middle attack, and most HTTP clients will refuse to connect. Public keys can be "revoked" by Certificate Authorities if the key is known to no longer be trustworthy - for example, if the associated private key was stolen. A certificate authority could revoke the public key of a malicious server owner, who could then no longer receive data from their malicious applications, as the HTTP client would refuse to connect.

The other case is the use of encryption other than HTTPS, such as AES, before the data is transmitted. This case is more plausible, but still unlikely at this time. Even among non-malicious applications, the rate of encryption use among Android applications is very low, and even those applications which do use it rarely use it correctly [6].

We have collected network traffic through multiple applications and capture methods to ensure that our method of traffic collection did not negatively influence the results. Additionally, by manually inspecting all captured traffic, we have attempted to verify that our tool correctly identified all cases of data exfiltration.

This method also has inevitable false positives: Some innocuous data will undoubtedly look like a telephone numbers or email addresses or locations. Additionally, the user filling out a web form is indistinguishable, from a network traffic perspective, from a malicious application sending sensitive data to a remote server. In order to alleviate these false positives, more work is needed to add "whitelists" of known-good servers, as discussed in the Future Work section.

VII. Future Work

In the future, we will extend our tool to recognize more types of personal data, including location data, calendar event information, and unique identifiers, such as IMEI number. Further, we will add a way for the user to import lists of information likely to stolen, such as contact lists, in order to reduce false positives.

Additionally, the traffic could be stopped to a remote server if it is being sent suspicious data until the user has had a chance to review the data and server in question. Approved servers could be saved in order to reduce future inconvenience to the user.

VIII. Conclusion

We have presented a method for using deep packet inspection to detect private data exfiltration in a platform-independent, general way which does not require the user to input personal data into the tool, extending beyond traditional data exfiltration detection approaches. We have demonstrated the effectiveness of this tool in detecting data exfiltration by malicious mobile applications, which can be put to use by end users to aid in detecting data exfiltration early, across multiple platforms.

References

[1] C. L. Abad and R. I. Bonilla. An analysis on the schemes for detecting and preventing arp cache poisoning attacks. In *Distributed Computing Systems Workshops, 2007. ICDCSW'07. 27th International Conference on*, pages 60–60. IEEE, 2007.

[2] G. Berry and R. Sethi. From regular expressions to deterministic automata. *Theoretical computer science*, 48:117–126, 1986.

[3] G. Combs. Wireshark. https://www.wireshark.org/, 1998-2016.

[4] T. W. D. Company. Circle with disney - internet. reimagined. parental controls and filtering. https://meetcircle.com/, 2015-2016.

[5] P. Developers. Privoxy. http://www.privoxy.org/, 2001-2016.

[6] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel. An empirical study of cryptographic misuse in android applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 73–84. ACM, 2013.

[7] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. Pios: Detecting privacy leaks in ios applications. In *NDSS*, 2011.

[8] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.

[9] D. Ferreira, V. Kostakos, A. R. Beresford, J. Lindqvist, and A. K. Dey. Securacy: an empirical investigation of android applications' network usage, privacy and security. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, page 11. ACM, 2015.

[10] ISO. 8601: Data elements and interchange formats – information interchange – representation of dates and times. Technical report.

[11] J. Kim, Y. Yoon, K. Yi, J. Shin, and S. Center. Scandal: Static analyzer for detecting privacy leaks in android applications. *MoST*, 12, 2012.

[12] A. Le, J. Varmarken, S. Langhoff, A. Shuba, M. Gjoka, and A. Markopoulou. Antmonitor: A system for monitoring from mobile devices. In *Proceedings of the 2015 ACM SIGCOMM Workshop on Crowdsourcing and Crowdsharing of Big (Internet) Data*, pages 15–20. ACM, 2015.

[13] B. LIU, J. Li, Y. LIANG, Z. Huang, X. Yang, et al. Smart pan provided with single temperature-sensing probe, and method for frying food, Oct. 8 2015. WO Patent App. PCT/CN2014/080,738.

[14] Y. Liu, C. Corbett, K. Chiang, R. Archibald, B. Mukherjee, and D. Ghosal. Sidd: A framework for detecting sensitive data exfiltration by an insider attack. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, pages 1–10. IEEE, 2009.

[15] A. Ornaghi, M. Valleri, E. Escobar, and E. Milam. Ettercap: A man-in-the-middle attack suite. https://ettercap.github.io/ettercap/, 2004-2016.

[16] M. Parkour. Contagio mobile mini malware dump, 2011-2016.

[17] D. C. Plummer. Ethernet address resolution protocol: Or converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware. STD 37, RFC Editor, November 1982. http://www.rfc-editor.org/rfc/rfc826.txt.

[18] A. Rao, A. M. Kakhki, A. Razaghpanah, A. Li, D. Choffnes, A. Legout, A. Mislove, and P. Gill. Meddle: Enabling transparency and control for mobile internet traffic.

[19] J. Ritter. ngrep: network grep, 2006.

[20] G. Shirts. Packet capture, 2015-2016.

[21] G. J. Silowash, T. Lewellen, D. L. Costa, and T. B. Lewellen. Detecting and preventing data exfiltration through encrypted web sessions via traffic inspection. 2013.

[22] S. Skafdrup, T. H. Sorensen, and H. Guld. Wireless thermostat with dial and display, May 4 2010. US Patent D614,976.