Security Testing for Mobile Applications

by Peter Kieseberg, Peter Frühwirt and Sebastian Schrittwieser (SBA Research)

In recent years, standard end-to-end encryption protocols have become increasingly popular for protecting the security of network communication of smartphone applications, as well as user privacy. On the whole, this has been a good thing, but this reliance has also resulted in several issues related to testing.

Software products have been becoming increasingly closely connected, particularly since the rise of mobile environments. Even programs doing menial tasks now require the user to go online, either to use cloud-based resources, or simply because the software provider does not charge users directly, but generates revenue by collecting and selling user preferences and advertising space. While the latter is often to be considered unethical, a study indicates that even those who complain are actually quite willing to give up privacy at a rather small price (see also for the ‘privacy paradox’ [1]).

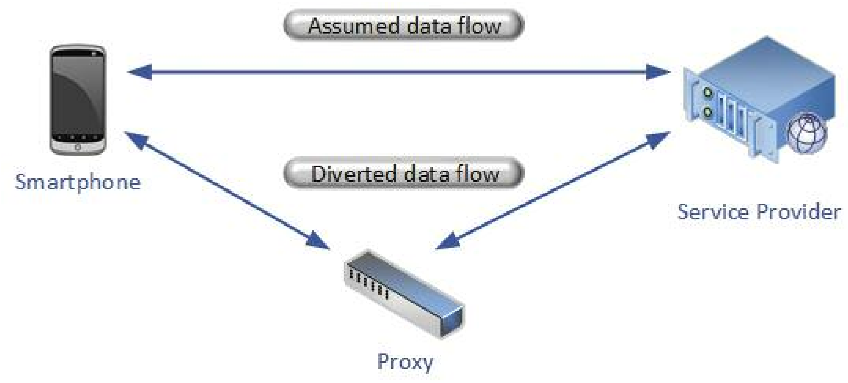

Figure 1: Testing approach.

This increase in connectivity is accompanied by a larger attack surface open to potential malicious actors. Furthermore, the time to market of software products is getting shorter as the market is getting increasingly competitive, resulting in a virtual omnipresence of security issues in current productive software. Even more problematic is the fact that most of the issues currently impairing software security are not based on new scientific findings or large-scale (criminal) organisations conducting intense research for new vulnerabilities to put to criminal use, but rather exploit known weaknesses introduced either by ignorance, or by putting trust into well-known security measures that were never designed for the problem at hand.

While the first problem refers to the problem of diminishing resources during the software development process and can be fixed using training, appropriate management and, most importantly, by setting aside resources, the latter is far more problematic and often touches at the very core of the design process of the software: Developers intrinsically assume certain aspects of their software and its use without further analysing the real attack surface. One very prominent example is the use of transport layer protection in mobile environments, with Transport Layer Security (TLS)[L1] being the de-facto standard for transport encryption. During a case study we analysed a set of mobile apps with respect to the usage of this protocol. We focussed on mobile environments since they possess some characteristics that are different from typical web-based applications. Condensing the results, we typically encountered two major issues with the use of TLS:

1. TLS was designed to provide secure end-to-end communication, always assuming the two endpoints to be trusted entities, i.e., removing them from the attacker model. Likewise, many software designers seem to assume that they possess full control over the client side of the software, which is clearly a problematic view in the case of mobile apps, where the client needs to be identified as a potential attacker. Many reasons for client side attacks can be found, e.g., in the realm of mobile messenger apps, users could try to impersonate other people in order to access or spoof messages. Particularly in apps that require the user to pay for the services provided, the motivation for attacking from the client side is rather straightforward.

2. Developers seem to rely far too much on the powers of TLS, even assuming capabilities, a protocol for transport layer protection can never hold up to. All TLS does is allow encrypted communication. ITLS cannot, for example, fix a protocol for authentication that is fundamentally flawed from a logical perspective. Nevertheless, we got the impression that in many apps TLS is seen as the silver bullet to fix any security-related issues.

These issues should typically be uncovered during the test phase, but our study on many popular apps, including mobile messengers, mobile games, social networks and even online ticketing shops, revealed an astonishing number of insecure protocols and a general misconception of the attacker surface. Our testing approach for uncovering insecure protocols (see [2] and [3]) relied on the fact that the smartphone the app is running on is under full control of us as adversaries, especially since any attacker having physical access to the phone can potentially manipulate hard- and software, including the operating system (or even run the whole app on a emulator without any actual smartphone). Since we controlled the smartphone we could make the phone route all of the (encrypted) traffic through a proxy, which was also controlled by us (see Figure 1). While TLS protected the communication from the phone to the proxy and from the proxy to the server of the service provider of the app, all the information is visible on the proxy, thus making every aspect of the protocols visible for analysis. For this approach, only standard tools are required, not only making this a good strategy for attacking apps, but also a viable and cost-effective measure for testing.

In conclusion, it is clear that the topic of software testing needs far more focus, especially considering the new dangers of fully integrated and interconnected environments as envisioned by ubiquitous computing and the 'internet of things'. Based on the experiments carried out in order to grasp the major problems and assess them, we will put our efforts into defining a testing approach suitable for interconnected environments, starting with the design phase of the software. Furthermore, we plan to support this approach with tools that will speed up the testing process for well-known and important protocols. Nevertheless, one of the major issues still prevalent with software testing, the issue of resources, cannot, in our opinion, be fixed on a technical level, but requires far more awareness on the management level.

Link:
[L1] https://tools.ietf.org/html/rfc5246

References:
[1] P. A. Norberg, D. R. Horne, D. A. Horne. "The privacy paradox: Personal information disclosure intentions versus behaviors", in Journal of Consumer Affairs 41, no. 1, 100-126, 2007.
[2] P. Kieseberg,et al.: "Security tests for mobile applications – Why using TLS SSL is not enough", in 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2015.
[3] R. Mueller, et al.: "Security and privacy of smartphone messaging applications", International Journal of Pervasive Computing and Communications, vol. 11, 2015.

Please contact:
Peter Kieseberg, SBA Research, Vienna, Austria
pkieseberg@sba-research.org

u'smile – Secure Mobile Environments

by Georg Merzdovnik, Damjan Buhov, Artemios G. Voyiatzis and Edgar Weippl (SBA Research)

Protecting user security and privacy on the internet takes more than transport layer security (TLS) and strong cryptographic algorithms: utilizing TLS notary services and certificate pinning for improved defences against prevalent third-party tracking.

Smartphones and mobile devices are becoming an integral part of life in the connected world. They allow easy access to information and content generation as well as consumption. A significant driving factor for their wide acceptance is the enormous number of available applications (commonly referred to as 'apps'). For example, by the beginning of 2017, the Google Play Store offered more than 2.6 million apps for smartphones, ranging from games and weather apps to office suites and banking apps. While these applications offer huge potential, they can also, either intentionally or inadvertently, pose a security and privacy risk to their users.

At the Josef Ressel Centre for User-friendly Secure Mobile Environments (u'smile) we are analysing security issues in current and future mobile applications. We are working on: (i) the design, development, and evaluation of concepts, methods, protocols, and prototype implementations for addressing security risks, and (ii) communication and co-ordination with industry partners and standardisation organisations with the goal of establishing globally accepted standards for secure, interoperable mobile services.

One area that we are working on is the security and privacy of network communications of mobile apps and their cloud-based services. All published applications go through inspection for malware behaviour before being published in the Google Play Store. However, user tracking and other forms of privacy leaks are still feasible, especially through advertisements included in apps and web applications to provide a revenue stream for the developers of free or low-priced software. A large number of tracker-blocking applications and apps are available for privacy-conscious users. We performed a large-scale study of the effectiveness of available tracker-blocking tools [1], which showed that despite their sophistication, tools cannot defend against all privacy leaks. Furthermore, a significant portion of applications transmit information over HTTP without establishing a secure connection over the TLS protocol (i.e., use HTTPS). This bad practice allows traffic interception and content inspection in-transit, increasing the threat of user tracking.

Digital certificates are used by TLS and allow verification of the identity of the server an app is connecting to as well as setting up a connection for exchanging information securely. Numerous certificate authorities come pre-installed in modern mobile operating systems, all of which are equally trusted by an app to offer a valid certificate for any internet server. TLS notary services collect and distribute information from TLS certificates presented at various points on the