2016 IEEE Symposium on Security and Privacy Workshops

Sensor-based Mobile Web Fingerprinting and Cross-site Input Inference Attacks

Chuan Yue
EECS Dept., Colorado School of Mines, USA

***Abstract*—Smartphone motion sensor data are not only accessible to native mobile apps, but have also become accessible to the webpages rendered in either mobile browsers or the WebView components of mobile apps. In this position paper, we highlight four types of broad and severe user fingerprinting and cross-site input inference attacks that can exploit the smartphone motion sensor data to compromise mobile web users' privacy and security; we also discuss some research topics for further investigating the effectiveness of these attacks and designing usable defense mechanisms.**

1. Introduction

Smartphones have become an indispensable communication and computation platform for billions of users. However, they have also been severely targeted by cybercrimes, and substantial benefits such as convenience and efficiency brought by them to users do not come without the high risks of security and privacy breaches. Especially, high-resolution sensors such as accelerometers and gyroscopes are often equipped in modern smartphones; they have enabled mobile apps to have richer functionality and better interactivity, but have also created many new opportunities for attackers to compromise users' security and privacy.

Sensor data are not only accessible to native mobile apps developed in specific programming languages such as Objective C for iOS and Java for Android platforms, but have also become accessible to the webpages rendered in either mobile browsers or the WebView components of native mobile apps along with the explosion of the mobile web and HTML5 adoption. Researchers have shown that sensor data can facilitate some types of smartphone security and privacy attacks based on the facts that manufacturing imperfections exist in hardware or the assumptions that malicious native mobile apps are installed (Section 2). However, many broad and severe security and privacy attacks that can occur to smartphone users ***due to the unrestricted motion sensor data access on webpages and per users' browsing behaviors*** have not been thoroughly examined yet.

In this position paper, we highlight four types of broad and severe user fingerprinting and cross-site input inference attacks that can exploit the smartphone motion sensor data to compromise mobile web users' privacy and security (Section 2). Here the ***mobile web users*** include mobile web browser users as well as mobile app users who interact with the web through the WebView components of apps; therefore, these attacks can potentially affect almost all the smartphone users. We also discuss some research topics for further investigating the effectiveness of these attacks (Section 3) and designing usable defense mechanisms (Section 4), and we sincerely welcome your suggestions.

2. The Four Types of Attacks

Smartphones often contain *motion sensors* such as accelerometers, gyroscopes, and compasses, *environmental sensors* such as thermometers and photometers, and *position sensors* such as GPS. We focus on investigating motion sensors because (1) unlike environmental sensors they often provide high-entropy data, (2) unlike position sensors they do not require apps including mobile browsers to obtain any special access permission, and (3) they are inherently pertinent to the behaviors of users. In recent years and on both iOS and Android platforms, browsers and WebView components have ***further extended the unrestricted motion sensor data access*** to regular webpages loaded on them. Basically, JavaScript code on regular webpages can register to receive *device orientation* events about the device rotation angles around the z, x, and y axes; it can also register to receive *device motion* events about the device rotation rates and device acceleration forces along the three axes.

Motion sensor data can be accessed by apps and by JavaScript code on webpages without requiring smartphone users to grant any special permission because they are not traditionally considered as sensitive. However, advanced attacks can exploit such "nonsensitive" motion sensor data to compromise mobile web users' security and privacy.

We define four types of advanced attacks as shown in Figure 1: (a) first-party user fingerprinting, (b) third-party user fingerprinting, (c) parent-to-child cross-site input inference, and (d) child-to-parent cross-site input inference. Fingerprinting attacks aim to compromise privacy, while cross-site input inference attacks aim to compromise security. It is important to note that ***no malicious app needs to be installed and no extra configuration or permission confirmation is needed*** to perform all these attacks.

2.1. User Fingerprinting Attacks

HTTP cookies are still widely used on both first-party and third-party websites to track users [14]. However, in recent years, many advanced tracking techniques (such as using supercookies, HTTP Etag, HTML5 local storage, and HTML canvas APIs [1], [3], [11], [12], [15]) have also become popular on the web.

DOI 10.1109/SPW.2016.17

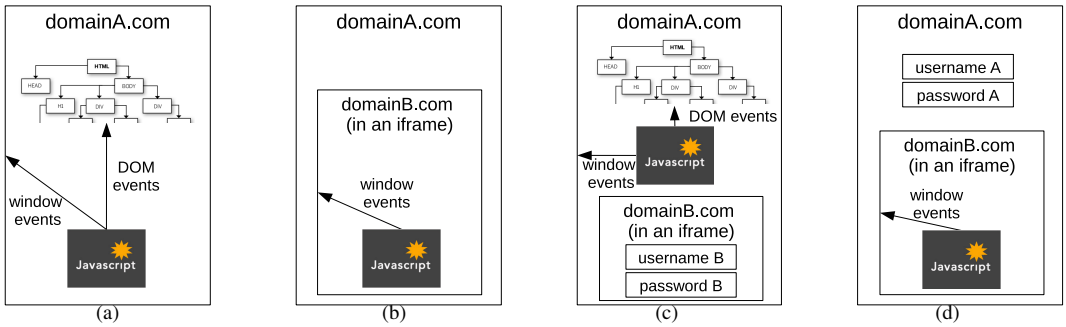

Figure 1. Sensor-based attacks: (a) first-party and (b) third-party user fingerprinting; (c) parent-to-child and (d) child-to-parent cross-site input inference.

Fingerprinting is the most challenging type of web tracking attacks. As analyzed by Eckersley in the Panopticlick study [8], for a user, avoiding being tracked (1) by basic stateful techniques such as HTTP cookies is tricky (e.g., need to configure the appropriate settings in browsers), (2) by advanced stateful techniques such as supercookies is harder (e.g., need to find ways to disable them), and (3) by stateless fingerprinting techniques will be most challenging. The Panopticlick study [8] is more about *browser fingerprinting* because the fingerprints are constructed based on the characteristics of the browsers; anonymous browsing solutions such as the Tor browser can potentially mitigate the risks of such attacks. Researchers have also investigated *smartphone fingerprinting* by exploiting the manufacturing imperfections in hardware [4], [6], [7]; calibration and obfuscation techniques can be effective in mitigating such hardware device fingerprinting attacks [4], [6].

Our fingerprinting attacks (Figures 1(a) and 1(b)) are different from and complementary to these existing attacks. ***Ours are more about user fingerprinting*** because fingerprints will be constructed based on the behaviors of users; therefore, they can be performed even across browsers and smartphones for the same user, and can be harder to be avoided without affecting the normal functionality of applications. ***The basic threat model*** for our user fingerprinting attacks is that a first-party website (whose domain is displayed in the browser's address bar) or a third-party website (whose documents are embedded, e.g., as *iframe* documents, in a first-party website) can exploit browsing behavioral biometrics obtained from motion sensors to track a smartphone user even if first-party and third-party persistent cookies are disabled, supercookies are removed, and browser & hardware device fingerprinting risks are avoided.

Specifically, in the first-party user fingerprinting attacks (Figure 1(a)), JavaScript code included or embedded in a first-party website can register to receive the window associated events for obtaining device orientation and motion data; although it can also exploit behavioral biometrics by monitoring DOM (Document Object Model) events, motion sensor data are more generally accessible and more representative of the characteristics of individual users. ***Even first-party user fingerprinting attacks can raise severe privacy concerns*** because a first-party website may either purposefully authorize a third-party website to learn about its users or accidentally allow a third-party website to do so due to insecure JavaScript inclusion practices, and users may not want to be tracked by a first-party website in the first place. Those are also the reasons why all the popular web browsers provide the privacy configuration features such as disabling first-party cookies and sending the "Do Not Track" requests to websites [22].

In the third-party user fingerprinting attacks (Figure 1(b)), JavaScript code in an iframe child document can register to receive the window events (from the window object of the iframe document) for obtaining device orientation and motion data, although it cannot access the DOM events of the first-party webpage due to the same-origin policy [21]. ***Such third-party user fingerprinting attacks can directly and severely compromise the privacy of mobile web users, and they can indeed be pervasively performed.*** For example, third-party advertisements are often included in iframes on millions of first-party websites. Malicious or compromised advertising websites [9], [20] definitely have the strong motivations to perform such attacks; legitimate behavioral advertising websites that infer user privacy for profit [16], [17] also have the strong motivations to do so.

2.2. Cross-site Input Inference Attacks

Researchers have shown that behavioral biometrics obtained from touch-screen and motion sensors can be exploited to infer smartphone users' sensitive inputs such as passwords [2], [5], [10], [13], [18]; however, these existing studies often assume that a malicious app is installed on a smartphone to perform the attacks, and they focus more on investigating attackers' capabilities of inferring the touch-screen lock PINs and passwords that could be valuable only if they are reused by the smartphone owner on some online services or if the smartphone itself is also stolen.

In our cross-site input inference attacks (Figures 1(c) and 1(d)), ***the basic threat model*** is that malicious JavaScript code can collect device orientation and motion data corresponding to soft-keyboard typing to build models for inferring users' sensitive inputs on cross-site web elements, regardless of the protection from the same-origin policy [21]. Note a parent document directly has the URL (context) information of its child documents, while a child (e.g., *iframe*) document can use the *document.referrer* value to obtain the URL (context) information of its parent document.

In the parent-to-child cross-site input inference attacks (Figure 1(c)), JavaScript code in a parent document can register to receive the corresponding window events for obtaining device orientation and motion data, from which it can further extract the portion of the data associated with the child window of the child document. The data for the child document can be feasibly extracted by the parent domain because keystrokes for an input field in a child document normally will not trigger temporally correlated DOM events for input fields in the parent document. The extracted data will be further analyzed to infer the individual keystrokes performed on the child document. ***Parent-to-child cross-site input inference attacks can cause severe consequences. A representative scenario*** is for insecure or even malicious Web Single Sign-On (SSO) relying party websites [19] to infer users' highly valuable SSO identity provider accounts (e.g., Gmail, Facebook, and Yahoo) typed in iframes.

The child-to-parent cross-site input inference attacks (Figure 1(d)) are similar to the third-party user fingerprinting attacks (Figure 1(b)) from the perspective of obtaining device orientation and motion data, but they have the different objective of inferring users' sensitive inputs on the parent document. ***Child-to-parent cross-site input inference attacks can directly and severely compromise the security of mobile web users, and they can be pervasively performed*** also due to the prevalence of using iframes to include advertisements into millions of first-party websites. Malicious or compromised advertising websites [9], [20] can be the main threat sources of such attacks.

3. Effectiveness of the Attacks

Our user fingerprinting attacks and cross-site input inference attacks can be modeled as multi-class classification problems. For the former, n users are n different classes with n unique fingerprints, and the problem is to identify if a current website visitor corresponds to one specific class or should be a new class. For the later, different soft-keyboard keys are different classes, and the problem is to identify the specific keys that are typed by a user on a soft-keyboard for sensitive web form inputs.

One common approach is to use machine learning algorithms to effectively perform the identification tasks: based on the collected motion sensor data, basic features such as acceleration forces and rotation rates will be extracted, and statistical features such as mean and standard deviation values will be derived; then, individual or ensemble-based machine learning classifiers will be trained.

In feature extraction, one main challenge is on segmenting (or aligning) the motion sensor data for individual user actions. Existing studies often simplify the segmentation task by using the ground truth tap (or key touch) events to align the corresponding motion sensor data [2], [10], [13], [18]. Unfortunately, such a simplification is not realistic for attackers because by default touch/tap events cannot be directly recorded either across apps or across origins on webpages. Cai and Chen briefly mentioned that they built a library of waveform patterns of keystroke motion to perform segmentation, but did not provide the detailed techniques and evaluation [5]. This main challenge is definitely worth further researching besides addressing other challenges such as determining the relevant features, appropriate classifiers, and optimal parameters for the four types of attacks.

Formal user studies should be performed to obtain the data for evaluating the effectiveness of our attacks. In terms of the user fingerprinting attacks, the overall entropy of the feature value distribution both between-subjects (for samples of different users) and within-subjects (for multiple samples of the same user across browsing sessions) should be calculated; a high between-subjects entropy and a low within-subjects entropy can to certain extent confirm that a feature is relevant. Similar to the Panopticlick study [8], it is desirable to compute the number of bits of fingerprint distribution entropy that can be achieved by our attacks.

In terms of the input inference attacks, the research can focus on detecting the cross-site context of the attacks and identifying essential procedures that can realistically lead to accurate data segmentation and keystroke inference. The following detection procedures are worth researching: (1) *device orientation detection*, i.e., accurately detecting whether the smartphone is in portrait or landscape mode, (2) *keyboard layout detection*, i.e., accurately detecting the type of the soft-keyboard in use thus the reference locations of the keys, (3) *input field detection*, i.e., accurately detecting the context of soft-keyboard typing such as the password or username field, and (4) *keystroke detection*, i.e., accurately detecting the keys that are typed by a user.

4. Usable Defense Mechanisms

One extreme is to completely block webpages' access to the motion sensor data, but this approach will immediately nullify the benefits of using motion sensor data in HTML5 for richer functionality and better interactivity. The other extreme is to always ask a user to grant or deny motion sensor data access requests on individual webpages, but this approach will not be usable or effective because users often do not pay attention to or do not understand permissions and often become habituated to granting permissions. Therefore, it is important to ***design fine-grained defense mechanisms that could be more usable and effective in practice***.

One potential mechanism is *element-based sensor data access control*. For example, *a new boolean attribute* can be added for HTML input elements, so that the access of motion sensor events can be disabled when a user is typing in (especially password type of) input fields. This fine-grained element-level access control mechanism can sufficiently protect against both parent-to-child and child-to-parent cross-site input inference attacks. It requires a browser or browser extension to support this new attribute by disabling the delivery of motion sensor events in appropriate time windows. It also needs individual websites to *opt in* to the protection by setting the attribute value. The opt-in approach will give websites the maximal adoption freedom with minimal compatibility issues. This new attribute can be further extended to HTML forms to make it convenient for

web developers. This mechanism is completely transparent to end users, thus having the obvious usability advantages.

The second potential mechanism is *frame-based sensor data access control*. For example, *a new value* for the iframe $sandbox$ attribute in HTML5 can be added, so that a child iframe document can access the motion sensor events only if this new attribute value is specified for the sandbox attribute by the parent document. This mechanism leverages the existing HTML5 sandbox mechanism to provide a fine-grained control of motion sensor events. It can sufficiently protect against third-party user fingerprinting attacks and child-to-parent cross-site input inference attacks that could be performed by an embedded iframe document loaded from a different origin. Similar to the first mechanism, it requires a browser or browser extension to support this new sandbox attribute value, and needs individual websites to *opt in* to the protection. This mechanism also has the obvious usability advantages due to its user transparency.

The third potential mechanism is *domain-based sensor data access control*. Similar to existing domain-based privacy and content settings in web browsers, this mechanism will take a *default-deny* or *default-allow* approach and then allow a user to define rules to *enable* or *disable* the access of motion sensor data for individual websites. This mechanism can sufficiently defend against all the four types of attacks including the first-party user fingerprinting attacks that cannot be defended by the first two mechanisms. A relatively simple support from a browser or browser extension is needed to implement this mechanism, but the main challenge is for users to become aware of this mechanism and properly use it. In other words, this mechanism is not transparent to users, and usability could be its main disadvantage.

The fourth potential mechanism is *domain and attack specific data perturbation*. Basically, a browser or browser extension first detects the specific attacks that may occur by analyzing the frame relationship and event registration activities (Figure 1), and then perturbs (e.g., adding noise to or decreasing collection frequency of) the sensor data that will be delivered to the corresponding receiver. The potential perturbation algorithms can leverage the results of the research on attacks (Sections 3), so that the effectiveness of the corresponding attacks can be decreased to the maximum extent without affecting the normal functionality of applications. This mechanism can be used to defend against all the four types of attacks, but its protection is statistical rather than deterministic as in the first three mechanisms. However, its obvious advantages are that it only needs to be deployed at the client side, and it is transparent to users.

5. Conclusion

We highlighted four types of sensor-based mobile web fingerprinting and cross-site input inference attacks, and discussed some research topics for further investigating the effectiveness of these attacks and designing usable defense mechanisms. We hope to raise researchers' and developers' attention to these attacks, and welcome your discussions.

Acknowledgments: We sincerely thank anonymous reviewers for valuable comments. This research was supported in part by NSF grants CNS-1624149 and DGE-1619841.

References

[1] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *ACM Conference on Computer and Communications Security (CCS)*, pages 674–689, 2014.

[2] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith. Practicality of accelerometer side channels on smartphones. In *Annual Computer Security Applications Conference (ACSAC)*, 2012.

[3] M. Ayenson, D. Wambach, A. Soltani, N. Good, and C. Hoofnagle. Flash cookies and privacy II: Now with HTML5 and ETag respawning, 2011. http://dx.doi.org/10.2139/ssrn.1898390.

[4] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh. Mobile device identification via sensor fingerprinting. *CoRR*, abs/1408.1416, 2014.

[5] L. Cai and H. Chen. On the practicality of motion based keystroke inference attack. In *International Conference on Trust and Trustworthy Computing (TRUST)*, pages 273–290, 2012.

[6] A. Das, N. Borisov, and M. Caesar. Tracking mobile web users through motion sensors: Attacks and defenses. In *Network & Distributed System Security Symposium (NDSS)*, 2016.

[7] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. In *Network & Distributed System Security Symposium (NDSS)*, 2014.

[8] P. Eckersley. How unique is your web browser? In *International Conference on Privacy Enhancing Technologies (PETS)*, 2010.

[9] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang. Knowing your enemy: Understanding and detecting malicious web advertising. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[10] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury. Tapprints: Your finger taps have fingerprints. In *International Conference on Mobile Systems, Applications, & Services (MobiSys)*, 2012.

[11] K. Mowery and H. Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In *Web 2.0 Security & Privacy (W2SP) workshop*, 2012.

[12] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *IEEE S&P Symposium*, 2013.

[13] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. Accessory: Password inference using accelerometers on smartphones. In *Workshop on Mobile Computing Systems & Applications (HotMobile)*, 2012.

[14] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.

[15] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle. Flash cookies and privacy. In *AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.

[16] K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, N. Provos, and M. A. Rajab. Ad injection at scale: Assessing deceptive advertisement modifications. In *IEEE S&P Symposium*, 2015.

[17] C. E. Wills and C. Tatar. Understanding what they do with what they know. In *ACM WPES Workshop*, 2012.

[18] Z. Xu, K. Bai, and S. Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WISEC)*, pages 113–124, 2012.

[19] C. Yue. The Devil is Phishing: Rethinking Web Single Sign-On Systems Security. In *USENIX LEET Workshop*, 2013.

[20] A. Zarras, A. Kapravelos, G. Stringhini, T. Holz, C. Kruegel, and G. Vigna. The dark alleys of madison avenue: Understanding malicious advertisements. In *Internet Measurement Conference*, 2014.

[21] SOP. https://www.w3.org/Security/wiki/Same_Origin_Policy.

[22] Tracking Protection. http://www.w3.org/2011/tracking-protection/.