Mobile Device Fingerprinting Considered Harmful for Risk-based Authentication

Jan Spooren, Davy Preuveneers, Wouter Joosen
iMinds-DistriNet
Department of Computer Science, KU Leuven
Leuven, Belgium
{jan.spooren, davy.preuveneers, wouter.joosen}@cs.kuleuven.be

ABSTRACT

In this paper, we present a critical assessment of the use of device fingerprinting for risk-based authentication in a state-of-practice identity and access management system. Risk-based authentication automatically elevates the level of authentication whenever a particular risk threshold is exceeded. Contemporary identity and access management systems frequently leverage browser-based device fingerprints to recognize trusted devices of a certain individual. We analyzed the variability and the predictability of mobile device fingerprints. Our research shows that particularly for mobile devices the fingerprints carry a lot of similarity, even across models and brands, making them less reliable for risk assessment and step-up authentication.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Authentication*

General Terms

Security, Measurement, Experimentation

Keywords

Authentication, Device Fingerprinting, Risk, Fraud Detection

1. INTRODUCTION

In recent years, the Internet has seen a steady growth and diversification in services offered, yet the predominant mode of authentication is still username and password authentication. This mechanism is often perceived as annoying by the user, because it tends to break the flow of the application and requires the user to remember or store a multitude of different strong passwords. Moreover, its strength is challenged by the advances made in brute-force and dictionary attack tools [15], combined with password databases leaked by hackers.

2015 European Workshop on System Security (EuroSec 2015)
Copyright 2015 ACM 978-1-4503-3479-2 ...$15.00.
http://dx.doi.org/10.1145/2751323.2751329

To limit the burden of authentication, a number of influential web sites such as Facebook and LinkedIn have therefore adopted an approach of using long-lived sessions, based on session cookies. Typically, these sessions can last for weeks and can be active concurrently (when a user is logged on using different machines – such as when using a desktop computer as well as a smart phone to access a web site). However, the gained usability of these long-lived sessions is offset to an increased risk of *session hijacking* [19], which requires additional mitigation to address this risk. For example, in the case of Facebook, users will typically be required to re-authenticate or identify friends in photos when the session is resumed from an unusual location.

Following this trend, there has been a recent interest in the Identity and Access Management (IAM) industry in using risk-based and context-based authentication [6] mechanisms for strengthening the level of trust in user sessions, thus enhancing usability by requiring fewer re-authentications. The concept is to use extra, 'contextual' information about the *session*, the *user* and his *device* to strengthen the level of authentication. The latter is typically utilizing browser fingerprinting (which is commonly perceived as a privacy threat [17]) for the 'benign' purpose of authentication. Browser fingerprinting has proven to be quite impressive in uniquely identifying a user [18, 8]: one could (and some actors do) believe that this technique can be seamlessly applied in the mobile ecosystem of smartphones and tablets.

In this paper, we present a critical assessment of the practice of using device fingerprinting for risk-based authentication purposes and formulate following conclusions:

1. We show that – contrary to web browsers on desktop and laptop computers – the fingerprints taken from mobile devices are far from unique.
2. We argue that the use of device fingerprint information does not constitute a suitable authentication mechanism, even if the fingerprints were unique.

In section 2, we review related work and the current state of practice in risk-based context-based authentication. We formulate our research hypothesis about the weakness of mobile device fingerprints in section 3, and discuss our approach on how we tested the hypothesis in a field study with participants fingerprinting their mobile devices. In section 4, we evaluate the results on the predictability of browser-based fingerprints and the study's validity threats; in section 5, we discuss mitigation techniques to strengthen risk-based authentication. Finally, in section 6 we present our conclusions on the practice of using device fingerprinting and propose directions for further research.

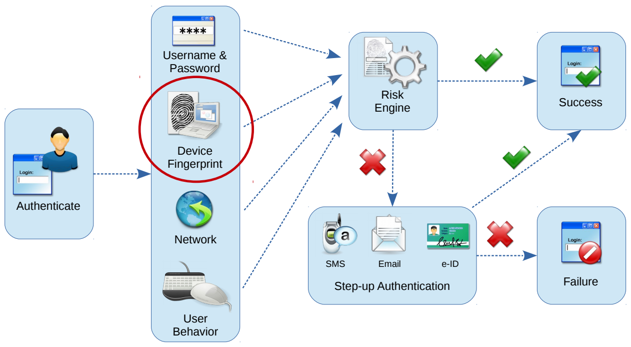
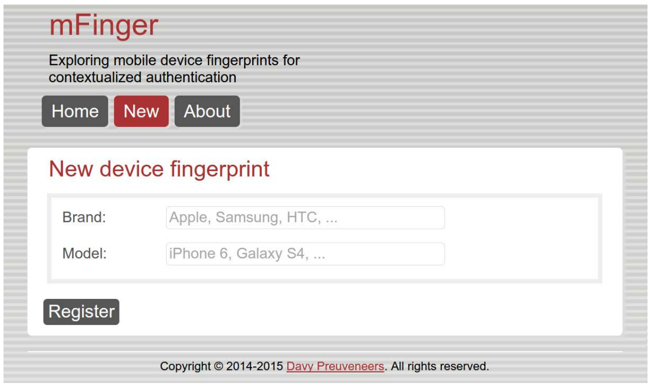


Figure 1: Risk-based step-up authentication



mFinger
Exploring mobile device fingerprints for contextualized authentication
Home
New
About
Register

New device fingerprint

Brand:	Apple, Samsung, HTC, ...
Model:	iPhone 6, Galaxy S4, ...

Copyright © 2014-2015 Davy Preuveneers. All rights reserved.

Figure 2: Collecting browser-based fingerprints on mobile devices with *mFinger*

2. RELATED WORK

Security risks with password-based logins have directed efforts in academia and industry towards looking for both stronger and more user-friendly authentication mechanisms. Initial efforts to aim for *zero-interaction authentication* [4] suggested the use of an authentication token to be worn by the user. However, practical and technical challenges have caused the idea to never really take off. It requires dedicated hardware to be worn by the user, as well as hardware to detect the presence of such an authentication token. Moreover, proximity-based authentication systems frequently suffer from vulnerability to relay-attacks [13, 2, 11].

Research into fingerprinting as a means of authentication was fueled by the Panopticlick [8] paper by the *Electronic Frontier Foundation*. They demonstrated the uniqueness of a browser among more than 4 million users who visited the Panopticlick website. Browser fingerprinting techniques are popular both with advertising and anti-fraud companies. Indeed, the FPDetective [1] project showed that about 1.5% of the top 10,000 websites is tracking individuals.

In the area of smartphones, various researchers explored the use of embedded sensors to uniquely identify a mobile device. In [5, 3], the authors investigated how the sensors on a smartphone can be used to construct a reliable hardware fingerprint of a phone. They analyzed whether the frequency response of the speakerphone-microphone loopback and the device-specific accelerometer calibration error have sufficient entropy to uniquely identify a device. Similar device fingerprinting and entropy analysis work was carried out with the digital camera [14, 12], the 802.11 wireless network [16] and acoustic background audio [21].

DARPA's Active Authentication program [10] explores different ways for a continuous authentication solution to make an informed decision on the identity of the user based on traits that can be observed through how people interact with the world (e.g. keystroke patterns, mouse and eye movements, cognitive aspects).

Similarly, the Identity and Access Management industry has been adopting context-based, risk-based authentication practices: Products such as *RSA SecureID* [9], *CA Risk Authentication* [7] and *PortalGuard* [20] have claimed support for this for a number of years now, although it is not always clear which properties are used to derive context from, nor which algorithms are used to build a trust score in an authenticated session. This makes evaluation of the strength of such *context- and risk-based authentication* solutions difficult.

3. HYPOTHESIS AND METHODOLOGY

While device fingerprinting appears to work relatively well for desktop machines and laptops, we suspected that it might be considerably less effective for mobile phones: Due to the *App Isolation* model used by most mobile phone and tablet operating systems, installing new apps will, for example, not change the font list available in the phone's web browser. For the same reason, mobile phone browsers usually do not feature a browser 'plug-in' model. Consequently, also the browser's *UserAgent* string will be defined mainly by the version of the browser.

In the remainder of this section, we will discuss our approach on how we investigated the hypothesis that browser-based device fingerprints are hardly unique and therefore a weak component for risk-based authentication in a more homogeneous ecosystem of mobile devices.

3.1 Approach

To test this hypothesis, we created an experimental setup using the *DeviceIdMatch* authentication module from ForgeRock's OpenAM 12 identity and access management system. This module is typically used used in an authentication chain with a *Risk Engine* and a *Step-up Authentication* system, requiring the user to authenticate with a stronger mechanism when the trust in the current session is deemed too low. This is illustrated in Figure 1.

To assess the adequacy of the fingerprint algorithm, we configured our own fingerprinting server – called *mFinger*[1] and shown in Figure 2 – with OpenAM's fingerprinting code. We invited a number of test users to visit the page using mobile phones and tablets. The test subjects were requested to enter their device brand and model and click a *Register* button. To prevent duplicate entries, the web page set a cookie so that multiple fingerprint submissions generated from the same device and web browser could be recognized as such.

3.2 Fingerprint collection

OpenAM's device fingerprinting technique is similar to the techniques used by advertisement brokers to track users across different web sites [18], where it is quite successful, in that such a browser fingerprint is unique enough to distinguish individual users with a high probability. Looking

[1]https://pang.cs.kuleuven.be/mfinger/

Fingerprint Component	# Distinct Values	Max. Identical Values
screen dim. + color depth	27	17
timezone	2	58
installedPlugins	4	50
vendor	4	40
installedFonts	7	37
browserLanguage	3	53
appMinorVersion	2	53
systemLanguage	3	53
cpuClass	2	53
userLanguage	3	53
userAgent	47	5
appName	1	59
appCodeName	1	59
appVersion	47	5
buildID	3	56
platform	7	42
oscpu	2	56
product	1	59
productSub	3	50
language	10	13

Table 1: Number of distinct values found per fingerprinting component, out of 59 records

closer at OpenAM's implementation reveals that it harvests a number of attributes from the browser to create a device fingerprint, ranging from the screen's resolution, the user agent of the browser, the language, the timezone, up to lists of installed plugins and supported fonts. The collected components are shown in the left column of Table 1.

3.3 Fingerprint matching

OpenAM's fingerprint matching algorithm is score and threshold-based. If a particular attribute in the fingerprint does not match, a configurable number of penalty points is added to a global score and if a certain threshold is passed, the matching fails. We noticed that some attributes are ignored in the matching algorithm. The algorithm only uses the following attributes, with in between brackets the penalty points in case of a mismatch:

- Screen dimensions and color depth (penalty points 50)
- Installed plugins (penalty points 100)
- Supported fonts (penalty points 100)
- Timezone (penalty points 100)
- User agent (penalty points 100)
- Geolocation (penalty points 100)

For a valid fingerprint only two attributes (the screen details and user agent) are mandatory, the other ones are optional. However, if a previously stored fingerprint contains a particular optional attribute, then follow-up submissions should include the attribute as well. Otherwise the comparison fails. Certain attributes, such as the timezone and screen details, require an exact match, whereas the user agent is first stripped from all version numbers before it is compared. The geolocation, if available, has an allowed matching range of 100 miles. For the plugin and font lists, up to 5 and maximum 10% of the entries can differ before the matching fails.

4. EVALUATION

In this section, we present a critical assessment of using device fingerprinting for risk-based authentication.

Brand	Occurrences	Models
Samsung	14	12
Apple	9	7
LG	7	2
Nokia	5	5
HTC	5	2
Motorola	5	1
Google	4	3
Sony	4	2
Other	6	5

Table 2: Mobile device brands and models

Model	Screen resolution
iPhone 4	480×320×32
iPhone 4s	480×320×32
iPhone 5	568×320×32
iPhone 5s	568×320×32
iPhone 6	667×375×32

Table 3: iPhone screen dimensions

4.1 Statistical breakdown of fingerprints

Out of the 69 respondents whose device fingerprints we collected during a period of one week, 1 duplicate fingerprint was removed, which had an identical cookie and was therefore a resubmitted fingerprint. Additionally, 9 records were removed, which appeared to be desktop or laptop computers, on which our study did not focus – leaving 59 records.

The brands of mobile devices detected and the different models encountered per brand are listed in Table 2. Table 1 lists the individual fingerprint components collected, as well as the number of unique values that were encountered (out of 59 fingerprinted devices) and the maximum number of identical values observed per component. We can see that, for example, for screen dimension and color depth, 27 different values were observed and the most popular value was shared by 17 different devices.

Looking at these findings, a number of components, such as *appName*, *appCodeName* and *product* appear not very useful: they returned the same value for all of our fingerprinted devices. The most useful fingerprinting components appear to be the *userAgent* and *appVersion*. However, the *appVersion* component is consistently identical to the *userAgent*, stripped of the `Mozilla/` head. Therefore, it does not introduce any additional entropy. In section 4.2, we take a closer look at the *userAgent* and *Screen dimensions* components.

4.2 Similarity of fingerprints

While a device fingerprinting technique is meant to uniquely identify a particular device, in the 59 collected fingerprints we found identical fingerprints for an iPhone 4 and iPhone 4s (except for the *language* component of the browser's *navigator* object – which is collected, but ignored by OpenAM's fingerprint matching) as well as for an iPhone 5 and 5s. The *userAgent* component of this duplicate fingerprint is shown here:

```
Mozilla/5.0 (iPhone; CPU iPhone OS 8_1_2 like Mac OS X) AppleWebKit/600.1.4
(KHTML, like Gecko) Version/8.0 Mobile/12B440 Safari/600.1.4
```

We also observed that the recorded iPhone 5, 5s and 6 products featured identical fingerprints (including the *userAgent* pictured above), except for the screen dimensions and the version numbers in the *userAgent* and *appVersion* fields. Both the screen dimensions (see Table 3) and version num-

Operating System	Font list
Android	cursive; monospace; serif; sans-serif; fantasy; Arial; Courier; Courier New; Georgia; Tahoma; Times; Times New Roman; Verdana;
iOS	cursive; monospace; serif; sans-serif; fantasy; Arial; Arial Rounded MT Bold; Courier; Courier New; Georgia; Papyrus; Times; Times New Roman; Trebuchet MS; Verdana;
Windows Phone	cursive; monospace; serif; sans-serif; fantasy; Arial; Comic Sans MS; Courier New; Georgia; Lucida Console; Tahoma; Times New Roman; Trebuchet MS; Verdana;

Table 4: Reported fonts per operating systems

bers are quite predictable and well-known.

We also recorded 5 Motorola Moto G devices: two of these devices featured a fully identical fingerprint. Another pair of devices was identical except for the (easily guessable and by OpenAM ignored) *navigator.language* string. Again, the difference between the *userAgent* and *appVersion* components in these two pairs was only in the version numbers. The two different *userAgent* components are shown here (different parts indicated by boxes):

Mozilla/5.0 (Linux; Android 4.4.4; $\boxed{\text{XT1032}}$ Build/KXB21.14-L1.$\boxed{\text{40}}$) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.89 Mobile Safari/537.36

Mozilla/5.0 (Linux; Android 4.4.4; $\boxed{\text{XT1039}}$ Build/KXB21.14-L1.$\boxed{\text{56}}$) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.89 Mobile Safari/537.36

Furthermore, 3 out of 4 HTC One S devices featured similar fingerprints: two devices fully identical, one device differing only in the *language* string, and 4 out of 5 LG Nexus 5 devices featured similar fingerprints: two were fully identical and two other differed only in the *navigator* object's *language* string. The *userAgent* and *appVersion* components were fully identical for these 4 devices.

Although the data collected was very limited, our hypothesis seems to be confirmed: For certain brands and models of mobile devices, the device fingerprint has so little entropy that the added value for its use in risk assessment is questionable.

4.3 Predictability of fingerprints

The above observations beg the question how much information is needed to actually fully reconstruct the complete fingerprint of a mobile device. Assuming that a user's device brand and model is known to an attacker; what is the probability of guessing the device's fingerprint? Looking at the individual components:

Screen dimensions and color depth: These attributes are fixed for mobile devices and can be looked up online in the hardware specifications. They are therefore fully predictable (100%)

Installed plugins: with the exception of 4 out of 41 Android devices and 6 out of 9 iOS devices that reported a Flash and/or Quicktime plugin, all of the other devices reported no plugins. The data from our limited study indicates a predictability of 88%.

Timezone and Geolocation: The timezone of a user is determined by her country of residence and therefore fully predictable (100%). Similarly, the geolocation would easily be guessed or obtained in a targeted attack.

For the **Installed Fonts**, we found a strong correlation with the browser/operating system. After clustering the supported fonts reported per operating system, we find that:

- Android: 93% of Android devices (or ≈ 69% of all mobile devices fingerprinted) used an identical set of 13 fonts
- iOS: All iOS devices fingerprinted (≈ 15% of all mobile devices fingerprinted) used an identical set of 15 fonts
- Windows Phone: All Windows Phone devices (≈ 12% of total devices) used an identical set of 14 fonts

See Table 4 for more details. As a result, if one knows the OS, one can predict the font list exactly in about 91% of the cases. Knowing the brand and model of a mobile device, typically implies knowledge of the operating system[2].

The set of **browserLanguage**, **systemLanguage** and **userLanguage** is only used on Windows Phone devices and are to a high degree interdependent. The **language** component is used by all devices, but is highly predictable in a targeted attack. The **vendor**, **appMinorVersion**, **cpuClass**, **platform** and **productSub** components are fully determined by the device brand and model and are fully predictable. However, none of these components are actually used by the OpenAM fingerprint matching algorithm.

The **userAgent** attribute is more complicated in the sense that its format is structured, but the device manufacturers or software providers can choose which content and fields they add. The structure in EBNF syntax is as follows:

```
user-agent       = server-val *(" " server-val)
server-val       = product / comment
product          = token ["/" product-version]
product-version  = token
comment          = "(" ctext ")"
ctext            = <any TEXT excluding "(" and ")">
```

Many mobile device brands include the device identifier as part of the user agent string. See the XT1032 Build/KXB21.14-L1.40 example used before, which can be traced back to a Motorola Moto G device. We collected about 6000 anonymous user agent strings for mobile devices from different log files. Several of these strings are identical except for some version numbers (OS or browser version, build number, etc.). We then used this list to check whether we could predict the *userAgent* attribute provided by the participant's mobile devices. Taking into consideration that OpenAM ignores version numbers by default during the fingerprint matching phase, the results are as follows:

- ≈ 41% of the user agents could be predicted when excluding version numbers
- An additional 24% could be predicted if the used browser is known up-front (some devices have multiple browsers)
- The remaining 35% of mispredictions were mostly due to unknown (rare or brand new) devices, or a different language combination (e.g. nl-be rather than en-us)

[2]We are assuming that the use of mobile OS replacements, such as CyanogenMod and Firefox OS is negligible.

In our assessment we did not take into consideration the popularity of mobile devices. We expect a higher prediction rate would be possible as the devices used by our participants is somewhat unbalanced (i.e. more uncommon devices such as smartwatches and Maemo/MeeGo mobile phone variants). Furthermore, the dataset we used to predict contained 6000 user agents of mobile devices (i.e. smartphones and tablets as classified with UADetector[3]). At *http://fingerbank.org* a more extensive dataset with devices and user agents is available, but it contains atypical user agent entries (such as strings starting with Windows Phone Search). These would need to be filtered before the dataset is used for prediction.

4.4 Validity threats

In this section, we briefly judge the quality of the empirical study and consider threats to the validity of the results in the previous section.

- *Conclusion validity*: In our field study, we only collected 59 mobile device fingerprints. Although this set contained many of the most common and popular devices, a larger sample set would have given us statistically more significant results.
- *Internal validity*: The fingerprinting and matching algorithms were based on a limited set of browser attributes. Some of these had a high predictability due to the localized target audience that participated in our study (e.g. the timezone attribute).
- *External validity*: We must be careful with generalizing our results and claiming the validity in other situations, as we chose a particular fingerprinting algorithm as provided by a contemporary identity and access management system, i.e. ForgeRock's most recent OpenAM 12 platform. Other fingerprint algorithms leverage more advanced components, such as HTML5 canvas fingerprints. A fairly extensive overview of fingerprints is available online at `http://BrowserSpy.dk`.

Even with the above reflections, there are a few other concerns to be taken into consideration why device fingerprints are not the holy grail for risk-based authentication. These are discussed in the next section.

5. DISCUSSION

In this section, we will provide some more general critical reflections about the use of browser-based fingerprints for risk-based authentication.

5.1 Attack model and threat analysis

Clearly, for mobile devices, device fingerprinting does not provide enough entropy to uniquely identify the device. However, the practice of using device fingerprints as an authentication mechanism is even more fundamentally flawed: it is built from information which can easily be obtained and then reproduced. It suffices for an adversary to lure a user into visiting a web page controlled by the adversary to obtain the browser's fingerprint. After this, impersonating a user by simulating the user's browser becomes trivial. The authentication mechanism is therefore reduced to the secrecy of the fingerprinting algorithm and of the individual components of the fingerprint, both of which can be obtained from a regular authenticated session with the web site under attack. It is therefore our opinion that using *device* context is not a good authenticator. At best, it can be used for 'negative authentication', i.e., for determining that a session might be compromised.

As mentioned, many of the fingerprints observed for mobile phones were identical. Where deviations within a particular mobile phone make or model were observed, the difference was typically in the version numbers stored in collected fingerprint components. Since, for example, the *userAgent* string is by default sent to all web servers, it suffices to have a typical web server log file to obtain all commonly used version numbers for a particular make and model of phone.

In practice, context-based authentication systems need some flexibility in the evaluation of version numbers, since devices get updated as part of their normal life cycle. In fact, the OpenAM script which compares fingerprints features a configuration setting `config.ignoreVersion` which causes the version numbers in the *userAgent* string to be ignored. This makes the situation even worse.

Since the *Panopticlick* initiative [8] by EFF, users have become more aware of web tracking and how it affects their privacy. Consequently, there is some pressure on web browser developers to make tracking users more difficult, such as by removing unnecessary sources of entropy in the browser or by deliberately introducing randomness in properties typically used for fingerprinting [17]. However, any initiative to make user tracking less effective will inevitably also make device fingerprinting less effective. An 'arms race' on fingerprinting techniques between developers of browsers or privacy-plugins and ad-broker companies would make this kind of authentication even more cumbersome, since the algorithms used would need to be updated constantly.

Even more troublesome is the use case of users who install plug-ins to deliberately confuse tracking strategies. These plug-ins would trigger web applications using 'context-based authentication' to always boost-up their required authentication mechanism (e.g., to text message based one-time passwords). For these users, 'context-based authentication' would imply a much less fluent authentication, entirely contrary to the goals and promises of context-based authentication.

5.2 Recommendations and road ahead

As mentioned in section 1, the concept of contextual authentication techniques is to use the extra 'contextual' information about the *session*, the *user* or the user's *device* for authentication purposes. In this paper we have been focusing on the device contextual information.

Using session context information, such as the used IP-address could marginally improve the confidence in the session, however *user behavior* fingerprints could be more useful: Contrary to device fingerprints, user behavior (i.e., the way the user operates the web site) is not as easily obtainable to adversaries as a device fingerprint. This would require behavioral fingerprinting, typically using unsupervised machine learning techniques, such as outlier detection.

Since different user behavior is not as apparent and as clearly detectable as a changed device, this type of fingerprinting would most likely require a Confidence function *C(s,t)*, modeling the level of confidence in the current user session s at time t. It seems natural to give this confidence function an exponential decay with a confidence half-life λ,

[3]http://uadetector.sourceforge.net

as long as non-typical behavior is recorded. A threshold confidence level would then be defined below which extra authentication would be required. Such a threshold level could be depending on the type of actions performed, requiring a higher level of confidence for actions that involve a higher cost or higher risk level. Investigating the feasibility of this technique is subject for future research.

6. CONCLUSION

In this work, we evaluated the device fingerprinting approach used by OpenAM version 12, focusing specifically on mobile devices. We found that the entropy of typical mobile device fingerprints is too small to be used for authentication purposes. The device fingerprints are quite predictable knowing the user's mobile device make and model. In fact, even with the limited number of devices investigated, several duplicate fingerprints were found. More problematic even is the fact that these fingerprinting components can be obtained easily by adversaries, making device fingerprinting unsuitable for authentication purposes.

Possibly, user behavior fingerprinting has more potential, since it is not as easily available to adversaries as are device fingerprints. In future work, we will further explore the applicability of behavior fingerprinting for context-based, risk-based authentication purposes.

Acknowledgment

This research is partially funded by the Research Fund KU Leuven.

7. REFERENCES

[1] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel. FPDetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*, CCS '13, pages 1129–1140, New York, NY, USA, 2013. ACM.

[2] Albert Levi, et al. Relay attacks on bluetooth authentication and solutions. *Lecture Notes in Computer Science*, 3280:278–288, 2004.

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

[4] M. D. Corner and B. D. Noble. Zero-interaction authentication. *MOBICOM'02 Proceedings of the 8th annual international conference on Mobile computing and networking*, 2002.

[5] S. Dey, N. Roy, W. Xu, and S. Nelakuditi. Leveraging imperfections of sensors for fingerprinting smartphones. *SIGMOBILE Mob. Comput. Commun. Rev.*, 17(3):21–22, Nov. 2013.

[6] N. N. Diep, S. Lee, Y. Lee, and H. Lee. Contextual risk-based access control. In S. Aissi and H. R. Arabnia, editors, *Proceedings of the 2007 International Conference on Security & Management, SAM 2007, Las Vegas, Nevada, USA, June 25-28, 2007*, pages 406–412. CSREA Press, 2007.

[7] P. Dulany, H. Gong, and K. Shah. 3D-Secure Authentication using Advanced Models. https://communities.ca.com/docs/DOC-231151630, 2014.

[8] P. Eckersley. How unique is your web browser? Technical report, EFF, 2009.

[9] EMC Corporation. RSA SecureID - Risk-Based Authentication. Data Sheet H13823. http://www.emc.com/collateral/data-sheet/h13823-ds-rsa-securid-risk-based-authentication.pdf, 2014.

[10] R. P. Guidorizzi. Security: Active authentication. *IT Professional*, 15(4):4–7, 2013.

[11] Z. Kfir and A. Wool. Picking virtual pockets using relay attacks on contactless smartcard. *SecureComm 2005*, 2005.

[12] C.-T. Li. Source camera identification using enhanced sensor pattern noise. *Information Forensics and Security, IEEE Transactions on*, 5(2):280–287, 2010.

[13] Lishoy Francis, et al. Practical relay attack on contactless transactions by using nfc mobile phones. *The 2012 Workshop on RFID and IoT Security (RFIDsec 2012 Asia)*, 2012.

[14] J. Lukas, J. Fridrich, and M. Goljan. Digital camera identification from sensor pattern noise. *Information Forensics and Security, IEEE Transactions on*, 1(2):205–214, June 2006.

[15] A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, pages 364–372, New York, NY, USA, 2005. ACM.

[16] C. Neumann, O. Heen, and S. Onno. An empirical study of passive 802.11 device fingerprinting. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 593–602, June 2012.

[17] N. Nikiforakis, W. Joosen, and B. Livshits. PriVaricator: Deceiving Fingerprinters with Little White Lies. In *Proceedings of the 24th International World Wide Web Conference (WWW)*, May 2015.

[18] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 541–555, Washington, DC, USA, 2013. IEEE Computer Society.

[19] N. Nikiforakis, W. Meert, Y. Younan, M. Johns, and W. Joosen. Sessionshield: Lightweight protection against session hijacking. In *Proceedings of the Third International Conference on Engineering Secure Software and Systems*, ESSoS'11, pages 87–100, Berlin, Heidelberg, 2011. Springer-Verlag.

[20] PortalGuard dba PistolStar, Inc. Contextual Authentication: A Multi-factor Approach. Tech Brief v.3.2-003. http://portalguard.com/pdfs/PG_CBA_Tech_Brief.pdf, 2012.

[21] Q. Quach, N. Nguyen, and T. Dinh. Secure authentication for mobile devices based on acoustic background fingerprint. In V. Huynh, T. Denoeux, D. H. Tran, A. Le, and S. B. Pham, editors, *Knowledge and Systems Engineering - Proceedings of the Fifth International Conference, KSE 2013, Volume 1, Hanoi, Vietnam, 17-19 October, 2013*, volume 244 of *Advances in Intelligent Systems and Computing*, pages 375–387. Springer, 2013.