Risk Based Web Authentication Using Bluetooth Devices

Asad Ali
Digital Identity and Security
Thales
Austin, USA
asad.ali@thalesgroup.com

Darmawan Suwirya
IBM Technology Services
IBM
Austin, USA
dsuwirya@ibm.com

Abstract - **There has always been a growing need for online access solutions that use strong authentication methods without encumbering the user. Current solutions that are easy to use and deploy offer weak security, while those that offer strong security are hard to use and deploy. The solution architecture presented in this paper allows users to continue using their existing authentication method, which in itself may be weak, but can be made stronger by augmenting it with standard Bluetooth devices that are part of the user's everyday work environment. These Bluetooth devices seamlessly offer additional factors of authentication without any explicit user intervention. This approach creates opportunities for adaptive, continuous and risk-based authentication where proximity of known Bluetooth devices is an input to risk management policies.**

Keywords - bluetooth devices; user convenience; adaptive online authentication; browser extension.

I. INTRODUCTION

"On the Internet, nobody knows you're a dog" [1] is an adage and meme about identity verification on the Internet, or rather lack thereof. It began as a cartoon caption by Peter Steiner and was published by The New Yorker on July 5, 1993. Ironically, a quarter of a century later, we still seem to be battling the same challenges of identity verification for Web applications, though on a different scale. Despite the collective desire of the security industry to get rid of knowledge-based authentication, passwords still remain the primary method of proving a person's identity on the Internet. This absence of progress has not been for lack of trying. There have been several approaches that augment the single authentication factor based on what-you-know, with a second factor based on what-you-have, or even a third one based on what-you-are. All these multi-factor solutions demand a dedicated hardware token. While these hardware tokens enhance the level of assurance associated with the authentication process, they also add complexity, thereby reducing usability, especially for millennials who are born and raised in the social media era and shun products that lack crisp user experience. For this reason, multi-factor authentication techniques are generally confined to controlled settings, such as enterprise and office environments where the possession of dedicated tokens can be made mandatory. However, the need for strong authentication is universal and is equally applicable in everyday Internet use outside these controlled environments.

This paper describes a technique of authenticating users based on their typical surroundings and work environment, by checking the presence of known or expected Bluetooth devices in the proximity of the user. The list of what devices to look for can either be explicitly specified by the user, or it can be implicitly learned by the system through previous successful logins.

The rest of the paper is organized as follows. Section II provides a background to the authentication challenge, and explains why the industry is still looking for a solution. Section III describes the detailed design, architecture and implementation of one such solution. Section IV offers an analysis of this proposed architecture in terms of security and user experience. We then offer our conclusions in Section V.

II. BACKGROUND

When using a Web browser to authenticate to an online service provider, a user typically enters his/her credentials into the Web page. In most cases, these login credentials are username and password. This single (what-you-know) factor is very easy to deploy, but universally considered weak. To improve the security of authentication, a second factor is added. This factor can be a smart card connected to the computer, a One-Time Password (OTP) generated on a dedicated device, a numeric code sent to the user's phone, etc. In all cases, the user has possession of this hardware token (what-you-have) that constitutes the second factor.

This second-factor hardware token has to be issued or configured by the service provider, thereby adding to the cost and complexity of deployment. The user cannot unilaterally select his/her own hardware tokens, such as one or more of the following devices: mobile phone, wireless keyboard, wireless mouse, wearable devices, wireless speaker, or any other Bluetooth enabled device and then use them as second-factor when authenticating online. These devices have to be issued by the Identity Provider. The reason for this restriction lies in the underlying authentication technology that demands a tight coupling of cryptographic algorithm between the client device and the backend authentication server. For example, the use of OTP or Public Key Infrastructure (PKI) requires both the device and the server to be updated together. How do the secret algorithm and the counter get exchanged between the device and the server? Will authentication flow use OATH Challenge-Response Algorithm (OCRA) [2], a challenge-response flavor of OTP? Similar complexities arise when using public key cryptography protocols.

The Fast Identity Online (FIDO) standard [3] addresses some of these issues by relaxing the tight binding between edge devices and backend authentication servers. However,

 ISBN: 978-1-61208-747-4

this adds a new constraint of only using devices that have FIDO stack built into them. The list of such devices, though slowly growing, is still a tiny fraction of the otherwise abundantly available Bluetooth devices.

There are additional usability concerns with existing approaches. For example, one-time numeric codes pushed to user's phone assume the phone is on the network and then also require manual action for the user to type the codes into a browser window. Biometric systems need specialized scanners and backend modifications. FIDO Universal Two Factor (U2F) [3] tokens require an action on the part of the user, such as pushing a button manually on the device.

Our solution enables an adaptive authentication framework so that the user's login is controlled by the presence of Bluetooth devices that can be preselected by the user. This authentication solution works without any explicit user action or interaction with hardware tokens, thereby addressing the following shortcomings identified in other existing approaches:

1. Devices for two-factor authentication have to be customized. They are also selected at the discretion of the authentication server or identity provider.
2. The deployment of these two-factor authentication devices is costly for merchants. Most often than not, this cost is indirectly passed to the user.
3. Users are required to carry these devices, which negatively impacts user experience.
4. The turn-around time to replace a device is long. User cannot select a new device unilaterally.

III. PROPOSED SOLUTION

This section describes the philosophy, architecture and implementation of the proposed solution.

A. Approach Overview

The solution presented in this paper can enable an adaptive authentication framework so that the user's login is controlled by the presence of Bluetooth or Bluetooth Low Energy (BLE) devices that are explicitly or silently selected at user's discretion, and not dictated by the Identity Provider. In addition to the primary authentication method (e.g., username and password), the presence of such previously identified Bluetooth devices is also checked. Virtually any Bluetooth device can be used based on user preference. Some examples include a user's mobile phone, wearable device (such as smart watch or fitness band), wireless keyboard, wireless mouse, wireless headphone, wireless speaker, smart pen, modern workout bench, weight scale, etc.

Risk based authentication already relies on data signals, such as network information, to confirm user's location, typing patterns to confirm user behavior, etc. Similarly, proximity of known Bluetooth devices can also indicate a trusted environment and hence allow for a more frictionless user login experience.

The challenge is to build this data signal handling into Web applications and enforce them through Web agents running on the user's computer. Web browsers can check the presence of Bluetooth devices in proximity by using Web Bluetooth Application Programming Interface (API) [5] or browser extension. One example of such Web browser extension is SConnect [4] that allows Web applications to seamlessly connect to security devices using a range of communication protocols, including Bluetooth.

B. High Level Design

The solution is largely based on common existing Web authentication system design, with slight enhancements added on both client and server sides in order to support the proximity of known Bluetooth devices as second factor authenticators. Figure 1 illustrates these components.

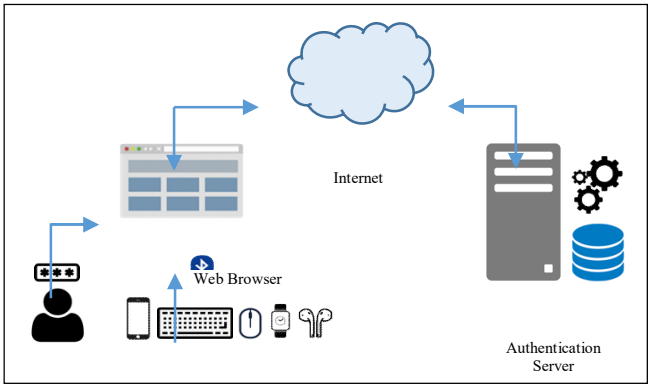

Figure 1. High-level block diagram of solution components.

These enhancements are simple to implement, yet powerful in their impact on delivering a seamless and secure authentication experience.

1) Client Modifications: On the client side, modification is done to the authentication page where the user would normally enter his/her username and password. The login page is augmented through a JavaScript code that uses Web Bluetooth API to silently scan and retrieve the list of Bluetooth devices found in proximity of the user's primary access device, for example his/her Personal Computer (PC). This API is currently only available in some modern browser versions. However, in the cases where Web Bluetooth API is not yet available on the user's browser, browser extension can be used to poly-fill the required functionality. This is standard practice in Web application development to compensate for missing browser capability by using an external library or component.

A list of the Bluetooth devices which are found in proximity of user, some minimal information about these devices, and the credentials entered by the user are then sent to the backend authentication server. The device information may contain its name, type, signal strength, etc. The user information can be as simple as user identifier, or could include actual credentials such as password, OTP or some other form of challenge-response.

 ISBN: 978-1-61208-747-4

Figure 2 shows a sample of code snippet that can be used for scanning Bluetooth devices in proximity using Web Bluetooth API specification. Once inserted into the login page, this script is expected to trigger itself when page is loaded. It makes the necessary API calls to instruct the Web browser to start scanning for Bluetooth devices. The script collects some minimal but useful information from every Bluetooth device detected in proximity of the device on which this Web browser is running. It then constructs the authentication request payload using this information. When the user has completed the login process, the login page along with this script is unloaded. Just before this unload step, the script can also do necessary clean up and instruct the Web browser to stop scanning for Bluetooth devices.

```
let btScan;
let btDevices = [];

// scan for BLE devices on page load
window.onload = () => {

 let bt = navigator.bluetooth;
 let opts = {acceptAllAdvertisements: true};

 btScan = bt.requestLEScan(opts).then(() => {

  let eventName = 'advertisementreceived';
  bt.addEventListener(eventName, event => {
   let device = event.device;
   btDevices.push({
    'name': device.name,
    'id': device.id,
    'serviceIDs': event.uuids,
    'appearance': event.appearance,
    'txPower': event.txPower,
    'rssi': event.rssi,
    'connected': device.gatt.connected
   });
  });
 });
};

// stop scanning for BLE devices on page unload
window.onunload = () => {
 btScan.stop();
```

Figure 2. Code snippet for scanning Bluetooth devices.

A sample of the authentication request payload could look like the data blob shown in Figure 3. The payload would still contain the user credentials as before, but in addition it will now also contain information regarding Bluetooth devices which are found in proximity. Examples of such information are device identifier (ID), device name, service IDs, appearance types, transmission power class, signal strength, and connection status. Individually each piece of information is minimal, but when used collectively, they are quite useful and unique enough to distinguish one device from another. Upon receiving authentication request payload, the backend authentication server then uses all this information to determine whether user is really who he claims to be.

A better alternative design could be to direct the client to look only for a predetermined collection of Bluetooth devices, guided through a set of high-level filter criteria. These criteria are determined by the server and passed to the client. The client then returns information related to devices that match these criteria. This design has several benefits, such as faster scanning time, collection of more relevant device information, and thereby smaller authentication request payload sent to the server. Furthermore, this approach is equally secure since it significantly reduces the risk of exposing policy or user-device mapping information.

```
GET /authenticate HTTP/1.1
{
  "username": "bob", "password": "1234",
  "devices": [

    {"id": "00:11:22:33:FF:EE",
     "serviceIDs": "0x2a01",
     "appearance": "0x10",
     "txPower": "4dBm",
     "rssi": "-40dBm",
     "connected": "false"},

    {"id": "00:12:23:34:AB:CD",
     "serviceIDs": "0x3440",
     "appearance": "0x20",
     "txPower": "0dBm",
     "rssi": "-52dBm",
     "connected": "true"}
  ]
}
```

Figure 3. Authentication payload sent by browser to server.

2) Server Modifications: On the server side, modification is done to the back-end authentication logic and infrastructure. The server needs to be able to accept additional payload in the authentication request. It then processes the information contained in this payload, including the information about Bluetooth devices. In addition to matching the username/password against user database, the authentication logic now also does verification of Bluetooth device information against the user/device database using policies that control how this new device information is expected to contribute to the final authentication decision. For example, one policy may require the presence of only one known device in the proximity of the user, while another policy may mandate that at least two such devices be present.

This processing not only yields a decision to grant or deny access to requested resource, but perhaps more importantly it also feeds the outcome into machine learning infrastructure that will make future authentication decisions smarter and more accurate.

C. Environment Initialization

Depending on the expected security level and type of authentication flow that is offered, the user may be requested to perform a set of minimal setups prior to using the system. Performing these optional setups has benefits. It increases the overall security of device selection and identification process. It also provides a fine grain control for the users to choose which devices they want to explicitly trust and use for this purpose. This is done through two sequences: device pairing and device binding. The user is free to choose either of these sequences or both of them, if so desired.

1) Device Pairing: As the name suggests, pairing means introducing two Bluetooth capable devices to each other.

After the pairing handshake, each device knows about the other. For example, pairing a Bluetooth headset with your smartphone or paring the smartphone with the infotainment system of your car. In the context of this paper user pairs one or more Bluetooth devices to his computer. This process is either done through tools available from the computer OS, or can be driven through the Web application itself with the help of either Web Bluetooth API or browser extension. This step is optional, but preferred as it increases trust level of the paired device, hence increasing the security of authentication. Only connection specific information is retrieved from the device. Device application related data such as workout history of a fitness band, are neither needed nor fetched during pairing. This can allay concerns about security and privacy of user data when such devices are used for two-factor authentication.

2) Device Binding: By "device binding" we mean the mapping of a user account to certain Bluetooth devices. This mapping is maintained on the authentication server. There are two ways authentication server can construct this map:

1. The user is asked to explicitly select devices.
2. The authentication server implicitly builds this list of devices over time, by monitoring available devices at the time of multiple successful logins.

In the first case, once the user has logged in to online service provider using an existing authentication method, the Web application will use the Web Bluetooth API or browser extension to look up for all Bluetooth devices found in the proximity and (optionally) paired to user's computer. The list of these available devices is shown to user in the Web browser. The user can now select one or more devices he wants to bind to his account. This information is stored in authentication server database and all subsequent access to user account will be granted only when the selected devices are also present at the time of authentication.

For risk-based authentication systems where explicit user involvement is not desired, or for user convenience, the system can decide not to expose user to device selection steps and can make device binding process transparent. The authentication server automatically builds the knowledge of which devices are relevant and associated to a user based on the device data silently collected over time, through multiple successful user authentications. For example, the user will never explicitly say "I want to associate my mobile phone and wireless mouse with my account". However, if on every login attempt the server notices that these two devices are present, it will implicitly bind these to the user account. After a certain number of such logins, as determined by the policy, device proximity factor can be enabled. At this point, the user can enter his login user identifier (e.g., email address) and authentication server will allow access without asking for any login credential, provided the same two devices are detected in the proximity.

D. Authentication Flow

This section explains detailed flows of authentication after the one-time environment initialization has been completed.

1) Adaptive Silent Authentication: In this flow, all required proximity devices are present, so the user can login simply by providing his/her user ID. No additional credentials such as password or OTP are required. This flow is illustrated in Figure 4, and outlines the steps taken by a user as well as authentication application from the initial intent by user to login to a Web resource, to the final action to grant access to the requested resource.

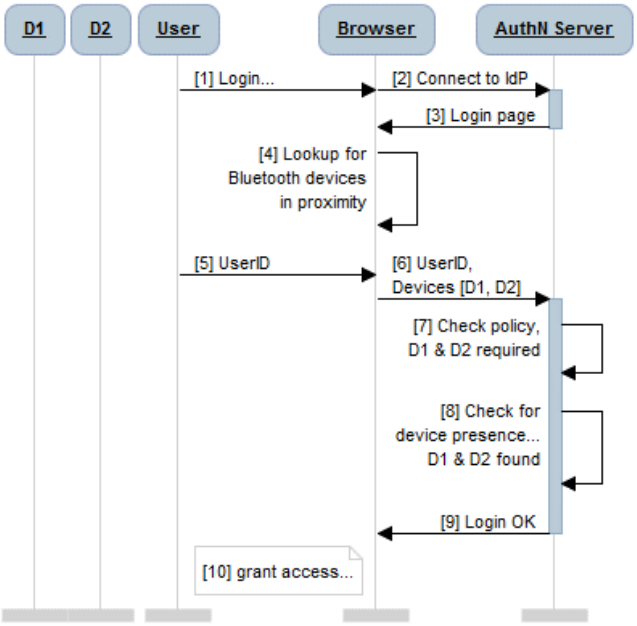

Figure 4. Adaptive authentication flow not requiring any user credential.

As shown in Figure 4, the user has two devices, D1 and D2, in close proximity during the login attempt. The numbered steps and message exchanges (1 to 10) are explained below:

1. The user initiates the login process by opening a Web browser from the access device such as a PC.
2. The Web browser client connects to the login portal of the authentication server: the Identity Provider.
3. The Authentication server sends back the login page to be rendered in the browser, asking the user to enter his *UserID*, such as username. This login page content also includes logic to scan for Bluetooth devices in the proximity of the user's computer.
4. This logic to scan for Bluetooth devices can be implemented by some JavaScript code that is automatically triggered by the Web browser when the login page is loaded. This code builds a list of devices it finds in the proximity of the user's computer. This is done either through direct use of the Web Bluetooth API, or through a browser

extension that invokes operating system level APIs to search for Bluetooth devices.
5. In parallel, the user enters his/her *UserID* (not the login credential, such as password) as required by the Web application. This information is entered within the same login page.
6. The Web browser sends this *UserID* to the authentication server, along with the list of Bluetooth devices found in proximity; in this case D1 and D2.
7. The authentication server looks up the login policy for this user, identified by *UserID*. It notices that the user had previously associated two Bluetooth devices with his/her account: a keyboard (D1) and a smartphone (D2).
8. The authentication server analyses the list of Bluetooth devices found in the proximity received earlier and verifies that both devices D1 and D2 are present.
9. The authentication server then determines that, since both devices are present, no further authentication credentials are required. It sends back a "Login OK" message.
10. User is granted access to the requested resource.

The user experience of this silent login flow is similar to password caching by the browser, or authentication server storing a cookie in the browser and automatically granting access to the user once that cookie is presented. However, the proposed Bluetooth device-proximity based logic offers a stronger assurance level since it relies on multi-factor authentication. In addition, it allows Web applications to create different policies to access resources of varying security levels, or when not all the expected devices are found in the vicinity of the access device.

2) Adaptive Step-up Authentication: An alternate flow can be when devices found in proximity of user are different from the expected list, thereby forcing the authentication server to ask for additional credential from the user. Depending upon the policy set by administrator of the resource being accessed, this credential could be a simple password, an OTP or even an elaborate PKI based challenge response through a dedicated security token. This step-up authentication flow is also triggered when no device is found in the vicinity.

3) Continuous Authentication: The Web application can optionally also perform continuous monitoring of the Bluetooth devices detected during initial login to make sure they remain in the proximity of the PC. In case they are removed, the application can be put in "stand-by" mode where further interaction is disallowed, until the missing device returns. Such continuous monitoring is possible through a browser extension. The user can now be logged in seamlessly without having to re-enter his primary credential. As an example of such continuous authentication, a user's Web session can be automatically put in "lock" mode when he/she moves away from his/her computer with device D2 (e.g. smartphone) in his pocket, and then restored when he returns to his/her computer.

IV. ANALYSIS OF PROPOSED SOLUTION

The design and architecture presented in this paper is a good compromise between security and usability. It is hard to find solutions that deliver on both. Most often, authentication solutions that are usable are not secure, while those that offer strong security generally do it at the expense of usability. Similar approach has been promoted in other reports [6] as well, further validating the value of using Bluetooth devices for authentication.

A. User Experience

Usability and user experience (UX) are gaining prominence in all security related products. This is not just for aesthetics or to merely appease users. Extremely robust and secure solutions that are not easy to use, end up degrading overall system security since users stop using them, and instead may find alternative options. The proposal identified in this paper is designed to improve UX without compromising security. It is modeled after "recognize" rather than ask for user credentials. The intent is to reduce user interactions with the authentication system. If a user connects from an office network, using his laptop, during office hours, and has his two previously identified Bluetooth device in close proximity, some application may consider this as sufficient authentication. Why bother the user for any login credential in this scenario? Some critical applications may additionally ask for only a single factor, like password. Still other highly sensitive applications could demand a higher assurance level of authentication like OTP or PKI based challenge-response from a dedicated hardware token. But this choice is based on the sensitivity of the Web application or the value of resource being accessed. In all cases, the surrounding environment acts as a silent "second-factor" to authenticate the user, thereby reducing the friction of login.

B. Security

As evident through numerous data breaches [7], attackers can easily compromise the what-you-know factor of a user's login credential such as password. If access to protected resources is controlled by this credential alone, an attacker can have full access to the resource. The promise of two-factor authentication (2FA) is to prevent attackers from having this free access using stolen or shared credentials. However, as mentioned earlier, the deployment and use of such dedicated tokens can be cumbersome for service providers and users alike. The silent use of Bluetooth devices serves the same purpose, but with far less overhead. As an attacker sends user's credential to the server, along with any Bluetooth devices in the proximity (if any), the server validates this credential and then matches the list of Bluetooth devices with the devices either explicitly bound to the account by user, or implicitly bound by the server based

 ISBN: 978-1-61208-747-4

on prior repeated successful authentications. Since the attacker does not have the specified configuration of devices, the login request is denied.

C. Policy

The proposed solution gives Web applications flexibility in deciding what authentication method is needed for which scenario. Similarly, the user is free to bind any number of devices to his/her account. In case multiple devices are bound to an account, the user or authentication server can specify whether all or a specific number of devices are mandatory for authentication. For example, a user binds his/her phone, his/her watch, and keyboard to his/her account, and specifies that any two of these three devices are needed for authentication. This rule is then enforced by the server when the user logs in.

Multiple valid login locations can be enabled with different configurations of known devices. In this way, a user may have a "work" or "home" configuration of specific Bluetooth devices that need to be present when logging to the account from these predefined locations.

D. Advantages

This adaptive and risk-based authentication using the proximity of user selected devices offers the simplicity of a single-factor knowledge-based authentication, but with the added security of multi-factor authentication. It also enables enterprise administrators to calibrate adaptive authentication with a range of policy options. Users will see these benefits:

1. User can continue to use the existing authentication method, e.g., password.
2. The second factor is based on devices picked by the user, not the device enforced by service provider.
3. Use of second factor is transparent, requiring little interaction or learning, especially when device use is automatically detected by server.
4. User can "walk away" from the computer and Web application is automatically put in stand-by mode.

Authentication servers will see these benefits:

1. Service provider can enforce strong multi-factor authentication at a very low cost of deployment.
2. There is no need to distribute dedicated 2FA hardware tokens to user. The user can use any existing Bluetooth token.
3. Service provider can offer adaptive authentication policies, based on resource being accessed.
4. Easy adoption by users means increased user base.

A key aspect of the approach in this paper is that Bluetooth devices can be accessed through a standard Web browser. There is no need to install a thick client on user's computer. Any Bluetooth device can be used, regardless of what application level software stack it supports. The only requirement is that the device advertises itself as a Bluetooth device. This approach is different from FIDO U2F devices, which rely on FIDO protocol stack in them. In our approach, any Bluetooth device of user's choice can be turned into a two-factor authenticator hardware token. Examples of such devices are: mobile phone, wearable devices like watch and fitness band, wireless keyboard, wireless mouse, wireless headphone, wireless speaker, smart pen, modern workout bench, weight scale, etc. The list is quite long, further validating the flexibility offered by this approach.

V. CONCLUSIONS

In the current era of digital transformation, all access control relies on some form of user authentication. This makes strong and context-based authentication an integral part of any system that protects resources and only grants access to authorized users. As the use of Internet and cloud-based services evolves, so does the expectation of "proper" authentication. Users now demand tailor made and fine-grained solutions that address their needs; nothing more, nothing less. They expect similar customization from authentication systems. In order to respond to this trend, identity providers will have to offer adaptive and risk-based authentication models. The goals should be to encumber the user with stronger authentication only if the resources being protected are worth the effort to use these stricter measures. The approach presented in this paper is one example of this adaptive and seamless authentication trend. Its efficacy can be further augmented by advances in artificial intelligence and machine learning.

ACKNOWLEDGEMENT

This research was done when the second author was working at Gemalto. Gemalto has since been acquired by Thales.

REFERENCES

[1] M. Cavna, "Nobody knows your're a dog: As iconic Internet cartoon turns 20, creater Peter Steiner knows the joke rings as relevant as ever", Washington Post, July 13, 2013. [Online, retrieved 10/2019] Available from https://www.washingtonpost.com/blogs/comic-riffs/post/nobody-knows-youre-a-dog-as-iconic-internet-cartoon-turns-20-creator-peter-steiner-knows-the-joke-rings-as-relevant-as-ever/2013/07/31/73372600-f98d-11e2-8e84-c56731a202fb_blog.html

[2] Internet Engineering Task Force, "OCRA: OATH Challenge-Response Algorithm", Specifications, [Online, retrieved 10/2019] Available from https://tools.ietf.org/html/rfc6287

[3] FIDO Alliance, Specifications, [Online, retrieved 10/2019] Available from https://fidoalliance.org/specifications

[4] D. Suwirya, H. Lu and L. Castillo, "Managing Access to Security Hardware in PC Browsers", Web Applications and Secure Hardware Workshop (WASH) pp. 3-9, London, UK, 2013, [Online, retrieved 10/2019] Available from http://ceur-ws.org/Vol-1011/1.pdf

[5] Github Repository. Web Bluetooth API, [Online, retrieved 10/2019] Available from https://webbluetoothcg.github.io/web-bluetooth

[6] B. Garska, "Two-Factor Authentication (2FA) Explained: Bluetooth Authentication" [Online, retrieved 2019] Available from https://blog.identityautomation.com/two-factor-authentication-2fa-explained-bluetooth-authentication

[7] T. Hunt, "The 773 Million Record Collection #I Data Breach", 2019, [Online, retrieved 10/2019] Available from https://www.troyhunt.com/the-773-million-record-collection-1-data-reach

 ISBN: 978-1-61208-747-4