Protecting Users from Compromised Browsers and Form Grabbers

Sirvan Almasi
Imperial College London
sirvan.almasi17@imperial.ac.uk

William J.Knottenbelt
Imperial College London
w.knottenbelt@imperial.ac.uk

***Abstract*—With the increasing use of the internet we are always in reach of a browser and a website. Users are continuously feeding sensitive information (such as passwords, personal and credit card information) to websites. These browsers and websites are susceptible to attacks from compromised client-side code, the browser and the operating system itself. Such threats emanate from Man-in-the-Browser (MitB) malware and form grabbers that are able to steal information from HTML forms and manipulate the forms at the cost of the user, resulting in the loss of sensitive information and financial costs. Whilst defensive techniques such as detection and prevention have their own merits, an out-of-band system can have superior security and user experience benefits. In this paper we explore the idea of circumventing the threats through a mobile phone-based system that can protect the user from compromised browsers and form grabbers. We build on the work of deeID, a blockchain-based and out-of-band identification and authentication system. Our contributions are the design of an out-of-band system dubbed FormL3SS, a standardised messaging for information request and the novel combination of existing techniques with a blockchain-based identity system. Our implementation of FormL3SS demonstrates the capabilities of sending data securely to a trusted server.**

I. Introduction

Form grabbers and Man-in-the-Browser (MitB) malware attacks are an effective method of stealing sensitive information and web traffic data. They can be used to steal credit card details, personal information and passwords, which are then either auctioned off [1] or directly used for fraud and the financial benefit of the attacker. Once the user's machine is infected, there is little modern browsers can even do. The Trojans and form grabbers act silently as the user interacts with the web page. One method of circumventing compromised browsers and operating systems is by using another device to submit the data (an out-of-band system).

In this paper we are concerned with a secure method of submitting data (typically done via a web form) by the user to the web server (which is hosting the website). We present a design, called FormL3SS, and a proof-of-concept implementation of an out-of-band form submission technique that can securely transfer required data (e.g. payment information) to the trusted server hosting the web application. We design and implement the proposed system on top of a blockchain-based identification and authentication system, deeID. deeID already offers a password-less and out-of-band authentication. Moreover, deeID offers the ability to send messages to the user via a secure communication channel.

Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb) 2020
23 February 2020, San Diego, CA, USA
ISBN 1-891562-63-0
https://dx.doi.org/10.14722/madweb.2020.23016
www.ndss-symposium.org

Form grabbers and MitB attacks are channelled through via various methods. In this paper we focus on two: Trojans and malicious third-party JavaScripts injected into the page.

Man-in-the-Browser (MitB) Trojans or financial malware silently infect a device and monitor outgoing communication before it is encrypted. Targeted Trojans attack online banking and e-commerce websites. ZeuS and SpyEye are two famous examples of such Trojans. Recent examples such as BackSwap [2] tailor their code to target specific banks, in this case Spanish banks.

Trojans are inherently form grabbers but in this context we will refer to form grabbers to scripts that infect third party libraries. It is common for most web applications to use third party libraries, mostly JavaScript, to enhance the functionality of the web page. Compromised libraries once loaded into a web page can scrape forms and steal information as they are entered into HTML forms. The most prominent example of such form grabbing attack is the British Airways' (BA) 2018 breach [3] where 380,000 customer payment details (including the CVV numbers) were stolen as they were typed into the page between Aug 21 and Sep 5. The attackers did not compromise the BA's servers but managed to compromise a third party library called `modernizr.js`. The attackers injected 22 lines of JavaScript code [4] that sent form data as they were submitted to a rogue phishing server.

The financial and economic cost of these attacks are to be taken seriously. Not only do the victims can face financial losses but services providers such as British Airways can also be fined for putting their customers in danger. BA faces a £183m fine by the UK regulators (ICO) [5].

With increasing number of individuals using the internet for online banking, shopping, etc. the pool of potential victims is only increasing in size. The importance of protecting users whilst enhancing their experience is evident and thus we shall propose a method to circumvent these attacks and allow the user to transact securely even if the browser or the operating system is untrustworthy.

A. *Contributions*

Our aim to contribute to the security and usability of web applications, especially when the application processes sensitive data.

- Design of FormL3SS: An out-of-band form and data transmission system.
- Implementation of the proposed system that can circumvent untrustworthy browsers and operating systems with respect to sending sensitive information to a trusted web server.
- Standardising form requests: We introduce a messaging standard that currently supports payment and sensitive information. Other type of information can be requested by signing the message's `form_type` as `custom`.
- Adding to deeID: deeID is a blockchain-based and out-of-band authentication system, we add an out-of-band form-submission functionality to it.
- Introducing the use of blockchain via deeID in managing secure interactions with the user and secure web servers.

B. *Paper Organisation*

We explain the basics and some history of Man-in-the-Browser threats in Section II. This section also includes an explanation of deeID - the blockchain identity and PKI system. In Section III we provide an overview of the proposed system and its key components. We then implement and show its working in Section IV. The process of sending data from the user to the trusted web server can be done via numerous methods; we explain them in Section V. In Section VI we discuss the system and its limitations. In Section VII we layout related literature. Lastly we conclude and provide future work in Section VIII.

II. Background

In this section we will go through Man-in-the-Browser threats, the decentralised PKI system (deeID) and out-of-band systems.

A. *Man-in-the-browser*

The primary focus of this paper is on Man-in-the-Browser (MitB) threats. MitB is usually a Trojan that infects a system and its applications (typically a browser). Once it infects a browser or an operating-system (OS) it can read network traffic, inject code into web pages, change HTTP headers, and generally manipulate the web page for the benefit of the attacker. MitB Trojans are also referred to as *Financial Trojans* because (typically) their aim is commit fraud and steal money from their victims. Such malicious actions are done by form grabbing banking details, personal details and changing transaction details (if the user is using online banking to transfer money).

One of the oldest and most famous MitB Trojan is ZeuS [6], also known as Zbot. First discovered in 2006, it has caused vast amount of financial damages. Its creator reportedly retired in 2010 and released [7] its source code to ZeuS' competitor, SpyEye [8].

Despite improvements in browser and OS security, criminals are finding creative ways in penetrating user systems in order to commit malicious activities. The British Airways example [3] in the introduction is a clear example.

Mobile phones are not safe either. A variant of ZeuS known as Zitmo [9] or The-Zeus-in-the-Mobile targets mobile phone devices.

B. *deeID System*

deeID [10] is a password-less and mobile phone-based identification and authentication system. It uses blockchain technology to create a network of unique identities for the participants. It also acts as a decentralised public-key infrastructure (PKI); allowing individuals to link public-keys to their identity. It has two main components: 1) Mobile phone application and 2) Blockchain contracts and functionalities.

The main use of deeID in this application is its identification and PKI tool. This allows us to independently verify a given domain and any content signed by that domain (public-keys associated with the domain. As you can see in Figure 1, the deeID contract has an array of public-key associated with the identity.

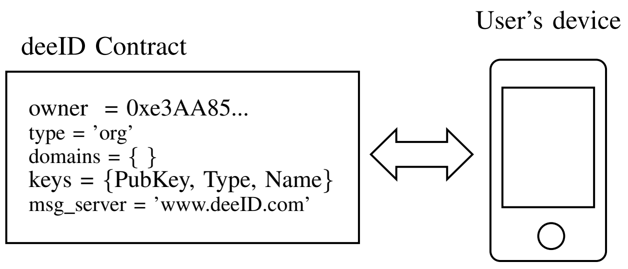


Fig. 1. Overview of the deeID system. Ownership of the deeID contract (identity contract) can be proven via private-keys stored on the user's mobile phone device.

The use of blockchain as a decentralised PKI is a new topic with ongoing research. There are numerous implementations and examples both in research and in industry. Our choice of deeID has been down to ease of development and the abstraction it provides. This is a replaceable component. Models of blockchain-based PKIs can be referred to in [11]; varieties of implementation include the likes of Namecoin [12], CertLedger [13] and Blockstack [14]. Other useful and related material include [15] [16] [17] [18].

C. *Out-of-band Transactions*

Out-of-band (OOB) authentication or transaction verification is a method whereby a secondary device and communication channel is used to carry out a process securely (such as authentication). Most of the literature is focused around authentication with examples including [19] [20]. Not all OOB authentication schemes are the same. Some are less secure than others, e.g. using SMS versus OOB push authentication mechanism; SMS can be subject to interception and redirection via SIM swap attacks.

III. Design

The goal of the proposed system, FormL3SS, is to circumvent compromised operating systems, browsers and malicious third party JavaScript libraries (form grabbers), thus creating a secure link between the user and the trusted server (the server hosting the website that the user is interacting with) in order to send sensitive data (e.g. payment information). We are countering two types of attacks: 1) Man-in-the-Browser Trojans such as *ZeuS* and *SpyEye* (also known as financial malware). 2) Form grabbers: Malicious third party libraries that scrape the web page and send the data to a controlled server. Malicious browser extensions fall between those two forms of attack. Therefore, we assume that all interactions and data on the web page displayed on the infected client can be manipulated and altered. The proposed system, FormL3SS, is shown in Figure 2.

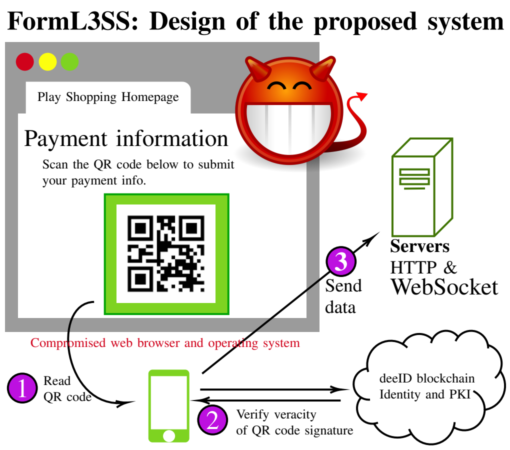

FormL3SS: Design of the proposed system
Play Shopping Homepage
Payment information
Scan the QR code below to submit your payment info.
Servers
HTTP &
WebSocket
3 Send data
Compromised web browser and operating system
deeID blockchain Identity and PKI
1 Read QR code
2 Verify veracity of QR code signature

Fig. 2. Visualisation of the proposed system. The web server communicates the required information (to be sent to the server by the user) to the client which is then displayed as a QR code. The mobile app scans the QR code, verifies the veracity of the signature. If verified, the mobile app sends the required data directly to the server.

Given that the user's computer (OS and browser) is assumed to be compromised we are using an out-of-band system to send sensitive information to the trusted web server. A mobile phone is a common device that has the ability to carry out the necessary cryptographic and networking operations. Thus, a mobile phone is used to send the data securely to the web server. The data from the client (web browser) is transferred to the mobile phone via a QR code. We have built on top of the deeID system because it already provides password-less and out-of-band authentication and thus this extends deeID for out-of-band transfer of other data.

We ensure that the QR code is not manipulated by the compromised browser and client-side code. We do this as follows: a) the trusted server (hosting the website) signs the data, b) the user scans the QR code, c) the user then types in the website's domain into the mobile phone app upon being asked for it, d) the domain names are cross checked with the scanned one from the QR code, e) the mobile phone app can now connect to the WebSocket server (given via the QR code) and ask for the authenticity of the signature, f) upon successful verification, the user will send the data. Asking the user to manually enter the domain name ensures there are no phishing attacks in play.

If the user has already registered with the website using deeID then the *link* created upon registration, we assume, is already persisted on the mobile phone. The mobile phone application will use this to guard against phishing attacks. The website's deeID blockchain contract should also contain the domains associated with the website's deeID which the mobile phone app can cross check with. This is shown in Figure 1.

Just like humans, fictional entities like websites can also have verified identities. This is most important if the website is supposed to have a legal identity. If the user has come across a website then they can attempt to get proof of identity of the website if another person or organisation has verified it. Then it is up to the user to trust the person or organisation that has verified the website's identity. This means that an attacker has to fake a web server, an identity contract and an identity proof to carry out a successful attack. These are strong barriers that the attackers have to overcome.

FormL3SS consists of the following components:

- Client-side or front-end code: JavaScript code is used to transform the data from the server into a QR image.
- WebSocket and server functions: A WebSocket connection is used so that the client can be updated upon confirmation (from the server) of receipt of the correct data from the mobile phone.
- Mobile app: We build on top of the deeID application to add out-of-band form transactions.
- Blockchain (deeID): In section V we will see alternative designs and solutions which will utilise the blockchain and deeID functionalities. We use the messaging function, proof of identity and its PKI functionalities.

A. Form types

For payment and personal information the server does not have to send a full HTML form to be filled in by the user. For payment and personal information, we have standardised it so that the server only has to identify a `form_type` in the QR code (or any medium of message). For payment the `form_type` is `card_info` and for personal information the `form_type` is `pers_info`.

For `card_info` the user will return an array containing the card type, number, expiry date and the CVV number. For `pers_info` the user will return an array containing the user's first name, surname, address (line 1, line 2, city, country and postcode), and date of birth.

Other forms will require a `form_type` being `custom`.

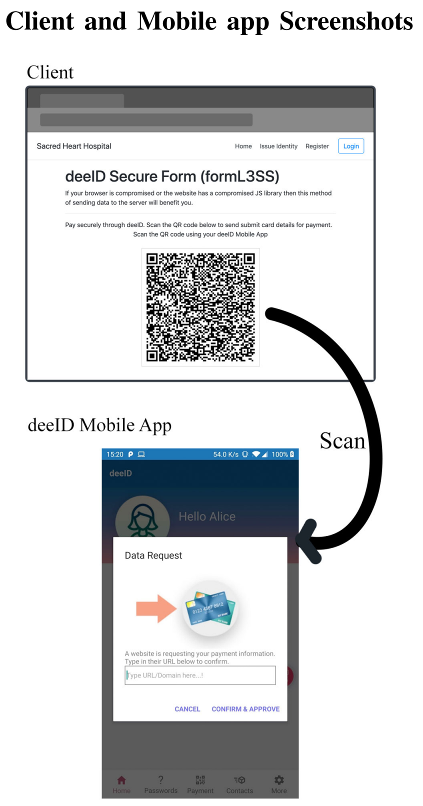
B. Custom form type

The custom form type indicated by `custom` currently supports the following input types: `text, number, email` and `date`. The advantage of this is that the added functionalities on deeID will do some form validation before being sent over to the trusted server. This will minimise the communication steps between the mobile phone and the trusted server.

IV. Implementation

We created a proof-of-concept that demonstrates the capabilities of the idea. The acting server was created using the `flask` [21] web framework. The front end of the dummy website used HTML and JavaScript. The front-end connected to another WebSocket library built on Python. The mobile phone functionalities extended those of deeID, with our proof-of-concept the mobile phone application can read a QR code that has a `type` called `deeIDForm`.

We will now go into further detail for the implementation and some of the data structures that are passed around. Figure 3 shows the interface of the acting payment web page and the mobile phone after scanning the QR image.



deeID Secure Form (formL3SS)

If your browser is compromised or the website has a compromised JS library then this method of sending data to the server will benefit you.

Pay securely through deeID. Scan the QR code below to send submit card details for payment.
Scan the QR code using your deeID Mobile App

A website is requesting your payment information. Type in their URL below to confirm.

Type URL/Domain here...!

Fig. 3. Interface design of the web page and the mobile phone app. Upon scanning the image the mobile app will ensure the QR code's data is not manipulated by verifying the domain and its cryptographic signature.

A. Back-end functionalities

The acting server (trusted web server) in our implementation picks up a user journey where the input of a user is required. In our context, we want the user to send a debit or credit card detail to the trusted server. Listing 1 shows the data structure sent from the server, it is requesting payment information from the user.

```
qr_msg = json.dumps({
        'type': 'deeIDForm',
        'domain': '',
        'uID': '', #Unique client & WS connection ID
        'form_type': 'card_details',
        'y': y, #Variable to link to a user session
        'exp_time' : '',
        'deeID': deeID, #Websites deeID address
        'sig' : str(sig), #Signature
        'ws_url': ws_url, #WebSocket URL
    })
```

Listing 1: The data behind the QR code which is signed and sent from the trusted server hosting the website.

B. Front-end functionalities

The front-end functionality is simply a QR code encoder. The client receives the data (signed) from the server and the JavaScript will create a QR image out of that data.

C. Mobile application functionalities

Upon reading the QR code from the compromised computer, the mobile application will attempt to verify the content and its signature. At this point the threat is that malicious actors could have a phishing server setup and have completely changed the content of the QR image to point to the phishing server. To mitigate this we ask the user to type in the domain of the site they have visited; then cross check it with the one in the QR image. The mobile app will then verify the public-key from the server.

Note, phishing threats are minimised through the deeID ecosystem if we have already established a connection with a service. This relationship is stored and any incoming communication is always verified with the stored information.

D. WebSocket functionalities

The purpose of the WebSocket server is to create a dynamic and user friendly application; that is to connect the client, server and the mobile phone. It is to allow each device to be updated upon state changes in other devices.

E. Custom form

We implemented a simple custom form functionality. This functionality allows the developer to request information that is not in the standardised form requests (currently payment and personal information). In the request data we have a `custom form_type`. This also requires another field named `form`; which is an array of the form fields required. As you can see in the Listing 2, we have requested a secret question and answer from the user. This then allows the mobile application to display the right form inputs as seen in Figure 4. This figure shows the two form fields requested by the trusted web server.

```
{
    'type': 'deeIDForm',
    'form_type': 'custom',
    'form': [
        ['text', 'Secret question', 'sec_ques_1'],
        ['text', 'Answer', 'sec_ans_1'],
    ],
    ...
}
```

Listing 2: Snippet of the QR code's data when a custom form is implemented. An extra `form` field to insert an array of required form fields.

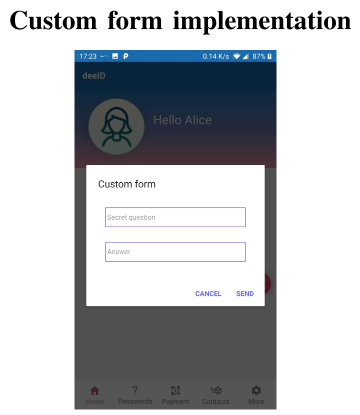


Fig. 4. Custom form interface design. Implementation example showing a server requesting two pieces of information.

V. Alternative Designs

In section III we described a simple design based on the objective of circumventing MitB threats and form grabbers. In this section we will explore different designs and architectures that have different benefits and can be more suited to a given web application. These alternative designs utilise more of the deeID functionalities: messaging server (being able to send messages to the user's mobile phone without using SMS), encryption and alternative keys stored on the user's deeID and the deeID mobile phone app itself.

The following factors determine the suitability of a design:

- Complexity of the form: QR code might not be suitable for a complex form.
- User and web server relationship: Is the user already registered with the trusted server?
- Whether the user has coupled a messaging server with their identity contract.

A. User is registered with the website and has a messaging server

If a user is already registered with the website of interest and they have a messaging server associated with their deeID contract then we can use this method.

Remark: We assume the user is able to use the password-less and out-of-band authentication method with the server.

The server, on request from the client, can send a secure message to the user's mobile phone application via their messaging server requesting a form to be filled in and sent back. Given that the user is already registered with the service they can verify whether it is a phishing threat (cross-checking domain, public-keys used to sign a message, etc.) and its veracity.

B. User is registered with the website and has no messaging server

Continuing from previous design but assuming there is no messaging server on the user's deeID contract. The trusted server (hosting the website) can encrypt the QR code data, send it to the client where it can be read by the deeID mobile phone application and be decrypted. From hereon the process continues as per the first design.

C. Send WebSocket server and request form from server

If the form (assuming a custom type form) is complex and thus cannot be fully transferred via a QR code; or perhaps the developer is looking for a more dynamic approach to form filling (e.g. the form is split into sections where an answer to one section will determine the type and content of consequent sections) then this method can be used.

Instead of sending all of the form in the QR code, we only send the URL of the trusted WebSocket server (belonging to the website of interest). Upon being verified by the user on their deeID mobile phone application, the user will begin requesting the form from the WebSocket server. At this point the process is a simple back and forth conversation between the two trusted devices.

VI. Evaluation and Discussion

Our objective has been to circumvent threats from Man-in-the-Browser Trojans, form grabbers and threats that fall in-between these two attack vectors. These types of attacks have been around for more than a decade and they are here to stay. They are getting far more sophisticated in targeting their victims and evading defences. Targeted attacks against banks and even cryptocurrency exchanges [22] using a mix of techniques are now becoming more common. MitB Trojans and form grabbers have primarily targeted web based payment services and hence commonly referred to as banking or financial malware. The primary goal of MitB and form grabbers is to essentially steal information that is being typed into a form or turn the user's actions against them (e.g. sending funds to the attackers bank account if the user is attempting to transfer funds).

It is clear that the threat from MitB and form grabbers is significant. The British Airways' case is evident to the scale of the threat. We can defend against these threats through preventive techniques that would guard the user's machine from being infected and detection techniques that would detect and eliminate the threat. However, our method has been to circumvent the threats, most helpful to a paranoid user. Moreover, whilst it is hard for the prevention and detection techniques to guard against form grabbers, our proposed system can defend the user from such a threat.

The proposed system, FormL3SS, is an out-of-band system that uses a mobile phone device to securely send data to the trusted server that is hosting the website.

Prevention and detection techniques have their own benefits. However, an out-of-band system can have superior user experience and security benefits. Typing in the same form data always detracts from the user experience of a website. With the proposed system that is combined with deeID the user can store information that is commonly required from websites, such as payment and personal information, on the device (in this case a mobile phone). This stored information can be requested with the proposed standardisation of requests; e.g. payment information is requested by a `form_type` being `card_info`. Standardising data requests is one of the contributions of this paper. Targeted information such as payment information can be standardised so that all devices know what data to send in order to satisfy the server without the server explicitly stating the input fields required. This reduces communication overhead.

Given that we have built on top of deeID which in itself is a form-less and password-less authentication scheme, the system can defend against phishing attacks too. Moreover, deeID provides an important role as identification has an important role when we are dealing with sensitive information online. We consider commonly shared sensitive information to be personal information (name, date of birth, address, etc.) and payment information. These are then interlinked with identity. Hence, the benefit of using deeID, an identity system with FormL3SS is that it can increase security with respect to identity theft and general credit and banking card fraud online. Providing the information and also proving ones identity on the fly adds an extra layer of security.

Malware attacks from banking to ransomware attacks are becoming more sophisticated. Collaboration between criminal gangs [23] and targeted attacks [24] enables better penetration and quicker responses to security guards being put up by individuals and organisations. IBM predicts [25] that with the emergence of cryptocurrency, criminal gangs are shifting to targeted ransomware attacks because they can evade the security forces and launder their money more effectively.

The proposed system has its own limitations too. Mobile phones are also susceptible to malware attacks. As the use of mobile phone devices grows they become more of an attractive target for malicious actors [26]. The fact that the user has to use an extra device for form submission is in itself an extra burden. Though, one can argue that because they are so common it is less of a burden. Also, because the user is not continuously typing in the same information, the process of submitting sensitive information is easier and more user friendly. The proposed system used deeID, an experimental technology. This brings about further complexities.

VII. Related Work

As far as we are aware, there are no other similar work where a general form submission protocol is used with the use of a mobile phone and a blockchain PKI system. However, there are similar work with respect to circumventing Man-in-the-Browser and compromised browser threats.

Software-based solutions that circumvent MitB threats include the likes of TrustJS [27] and ShadowCrypt [28]. TrustJS enables trust-worthy execution of JavaScript inside a browser by using trusted hardware (Intel SGX in their case) and a browser extension. This enables client-side form input validation. ShadowCrypt enables encrypted input/output and runs as a browser extension. Whilst it is successful in preventing the DOM containing sensitive data and thus being accessible to other extensions and compromised libraries, it doesn't prevent key loggers and other sophisticated Trojans. Privly [29] performs similar functions to ShadowCrypt but also uses a third party server for encrypted data management.

The work of Saba Eskandarian et al. [30], dubbed Fidelius, uses additional hardware to completely secure the I/O. In Fidelius the entire I/O path is protected by having a trusted dongle between monitor/keyboard and the compromised computer. The drawbacks of Fidelius are page load times and display response latency; as the additional hardware and operations from the keyboard and the display effect the user experience. We think that our proposal enhances the user experience by eliminating the input of data that is already securely stored on the mobile phone. Also, given that nothing is required on the browser there is no impact on page load and response rates. Bumpy [31] is similar to Fidelius that can be considered a predecessor to Fedelius.

VIII. Conclusion

In this paper we have discussed the threats from Man-in-the-Browser malware and form grabbers, highlighting the importance of these threats with recent examples. We propose a design to protect the user from compromised browsers and client-side or front-end code.

We have designed an out-of-band solution using a mobile phone, named FormL3SS. This system can circumvent MitB and form grabbers safely while improving the user's experience whilst interacting with the website. We implemented this design on top of an out-of-band authentication and blockchain-based system, deeID. We believe the combination of technologies used is a novel contribution. Moreover, a first in proposing a standard messaging format for out-of-band form submission systems. Developers can request payment and personal information without explicitly defining the form; this reduces communication overheads. Other data can be requested using a `custom` tag.

Future work include: website identification using deeID, to add an extra level of security; refining the messaging standard between the trusted server and the user's mobile phone; implementation of the alternative designs that were proposed; and implementing the encryption functions for the transmission of the data.

Acknowledgement

The authors would like to thank the anonymous reviewers for their helpful comments.

References

[1] Benoît Dupont, Anne-Marie Côté, Jean-Ian Boutin, and José Fernandez. Darkode: Recruitment Patterns and Transactional Features of "the Most Dangerous Cybercrime Forum in the World". *American Behavioral Scientist*, 61(11):1219–1243, October 2017.

[2] Limor Kessem. BackSwap Malware Now Targets Six Banks in Spain. https://securityintelligence.com/backswap-malware-now-targets-six-banks-in-spain, August 2018.

[3] How Hackers Slipped by British Airways' Data Defenses | WIRED. https://www.wired.com/story/british-airways-hack-details.

[4] The British Airways Breach: How Magecart Claimed 380,000 Victims. https://www.riskiq.com/blog/labs/magecart-british-airways-breach.

[5] British Airways faces record £183m fine for data breach. https://www.bbc.com/news/business-48905907, July 2019.

[6] Trojan.Zbot | Symantec. https://www.symantec.com/security-center/writeup/2010-011016-3514-99.

[7] Zeus and SpyEye hybrid emerges. *Network Security*, 2011(2):20, February 2011.

[8] Trojan.Spyeye | Symantec. https://www.symantec.com/security-center/writeup/2010-020216-0135-99.

[9] Najla Etaher, George R.S. Weir, and Mamoun Alazab. From ZeuS to Zitmo: Trends in Banking Malware. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 1386–1391, August 2015. ISSN: null.

[10] S Almasi and W J.Knottenbelt. Human identity and the quest to replace passwords continued - manuscript submitted for publication. 2019. Manuscript Submitted for Publication.

[11] Artem S. Konoplev, Alexey G. Busygin, and Dmitry P. Zegzhda. A Blockchain Decentralized Public Key Infrastructure Model. *Automatic Control and Computer Sciences*, 52(8):1017–1021, 2018.

[12] Namecoin. https://www.namecoin.org.

[13] Murat Yasin Kubilay, Mehmet Sabir Kiraz, and Hacı Ali Mantar. CertLedger: A new PKI model with Certificate Transparency based on blockchain. *Computers & Security*, 85:333–352, August 2019.

[14] Blockstack. https://blockstack.org.

[15] L. Dykcik, L. Chuat, P. Szalachowski, and A. Perrig. BlockPKI: An Automated, Resilient, and Transparent Public-Key Infrastructure. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 105–114, November 2018.

[16] A. Yakubov, W. Shbair, and R. State. BlockPGP: A Blockchain-Based Framework for PGP Key Servers. In *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*, pages 316–322, November 2018.

[17] E. Karaarslan and E. Adiguzel. Blockchain Based DNS and PKI Solutions. *IEEE Communications Standards Magazine*, 2(3):52–57, September 2018.

[18] Christos Patsonakis, Katerina Samari, Aggelos Kiayias, and Mema Roussopoulos. On the Practicality of Smart Contract PKI. *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, pages 109–118, April 2019. arXiv: 1902.00878.

[19] Longfei Wu, Xiaojiang Du, Wei Wang, and Bin Lin. An Out-of-band Authentication Scheme for Internet of Things Using Blockchain Technology. In *2018 International Conference on Computing, Networking and Communications (ICNC)*, pages 769–773, March 2018.

[20] Hirotaka Nakajima, Shigeya Suzuki, Tetsuro Tokunaga, Kiyoshi Tanaka, Yasuhiko Miyazaki, Koichi Maruyama, and Osamu Nakamura. Out-of-band authentication protocol for digital signage and smartphone interaction. In *2016 IEEE 5th Global Conference on Consumer Electronics*, pages 1–2, October 2016. ISSN: null.

[21] Flask. https://palletsprojects.com/p/flask.

[22] Ruby Cohen Moshailov, Roy. Gozi Adds Evasion Techniques to its Growing Bag of Tricks. https://www.f5.com/labs/articles/threat-intelligence/gozi-adds-evasion-techniques-to-its-growing-bag-of-tricks.html, January 2019.

[23] The Business of Organized Cybercrime: Rising Intergang Collaboration in 2018. https://securityintelligence.com/the-business-of-organized-cybercrime-rising-intergang-collaboration-in-2018, March 2019.

[24] Remi Sara Boddy Cohen, Roy Moshailov. Gozi Banking Trojan Pivots Towards Italian Banks in February and March. https://www.f5.com/labs/articles/threat-intelligence/gozi-banking-trojan-pivots-towards-italian-banks-in-february-and-march.html, April 2019.

[25] IBM X-Force Security Predictions for 2020. https://securityintelligence.com/posts/ibm-x-force-security-predictions-for-2020, December 2019.

[26] Christian Szongott, Benjamin Henne, and Matthew Smith. Evaluating the threat of epidemic mobile malware. In *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 443–450, October 2012. ISSN: 2160-4886.

[27] David Goltzsche, Colin Wulf, Divya Muthukumaran, Konrad Rieck, Peter Pietzuch, and Rüdiger Kapitza. TrustJS: Trusted Client-side Execution of JavaScript. In *Proceedings of the 10th European Workshop on Systems Security*, EuroSec'17, pages 7:1–7:6, New York, NY, USA, 2017. ACM. event-place: Belgrade, Serbia.

[28] Warren He, Devdatta Akhawe, Sumeet Jain, Elaine Shi, and Dawn Song. ShadowCrypt: Encrypted Web Applications for Everyone. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1028–1039, New York, NY, USA, 2014. ACM. event-place: Scottsdale, Arizona, USA.

[29] Privly. https://priv.ly.

[30] Saba Eskandarian, Jonathan Cogan, Sawyer Birnbaum, Peh Chang Wei Brandon, Dillon Franke, Forest Fraser, Gaspar Garcia, Eric Gong, Hung T. Nguyen, Taresh K. Sethi, Vishal Subbiah, Michael Backes, Giancarlo Pellegrino, and Dan Boneh. Fidelius: Protecting User Secrets from Compromised Browsers. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 264–280, May 2019. ISSN: 1081-6011.

[31] Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Safe Passage for Passwords and Other Sensitive Data. In *NDSS*, 2009.