# tiqr security audit

version: 1.1
date: November 2011
authors: Roland van Rijswijk and Joost van Dijk

summary: In May 2011 SURFnet hired Madison Gurkha to perform a security audit on the tiqr application for iOS and Android as well as for the server-side reference implementation. This report contains a summary of the findings of this security audit and details how the vulnerabilities that were identified in the security audit have been mitigated in the current tiqr release. It includes a summary of the results of a re-audit performed by Madison Gurkha in September 2011 on the updated tiqr application and server-side reference implementation.

# Table of contents

# 1 Introduction

## 1.1 Purpose of this report

In April of 2011 SURFnet released the first version of tiqr for iPhone to the public in the App Store, soon followed by the Android version. At the same time, a server-side reference implementation was made available through the tiqr website (https://tiqr.org/).

To ensure that tiqr meets security expectations SURFnet hired external security auditor Madison Gurkha and asked them to perform and extensive security audit. This report contains a summary of the findings from the security audit and details how these findings lead to changes in the tiqr applications to mitigate the issues that were identified. Parties interested in reading the full report should contact the tiqr team via tiqr@surfnet.nl.

The publication of this document was approved by Madison Gurkha.

## 1.2 Reading guide

This document is organised as shown in the list below:

- Chapter 2 introduces the goals set out for the security audit and gives a high-level overview of the methodology that Madison Gurkha used to perform the audit.
- Chapter 3 describes the issues identified by the auditors and documents the changes that were made to the tiqr application and server to address the issues described.
- Chapter 4 concludes the document with recommendations for organisations wishing to deploy tiqr in production.

## 1.3 Revision history

| Version | Date | Author | Changes |
|---|---|---|---|
| 0.1 | 2011-08-19 | Roland van Rijswijk | Initial version derived from full security audit report |
| 0.2 | 2011-10-27 | Roland van Rijswijk | Document updated after re-audit by Madison Gurkha |
| 1.0 | 2011-11-07 | Roland van Rijswijk | Minor rework after internal review |
| 1.1 | 2011-11-23 | Roland van Rijswijk | (post publication) fixed some mistakes reported by Brook Schofield of TERENA |

# 2 Goals and methodology

## 2.1 Goals for the security audit

The initial design concept for tiqr was conceived of at SURFnet in late 2010. This design concept consisted of:

- A storyboard description of the user workflow
- An architectural design describing the client and server roles
- A simple cryptographic security concept based on OCRA

Based on this initial design Egeniq developed a proof-of-concept implementation code-named “Moby Dick”. This proof-of-concept proved to be very successful; SURFnet demonstrated the technology to its constituency and got positive feedback. An article on popular Dutch technology forum “Tweakers.net” also resulted in positive reactions from a wider audience.

Because of the positive feedback and because an internal business case existed for a strong authentication technology that would be easy to deploy, SURFnet decided to create an open source product based on the proof-of-concept. The results of this project are the tiqr mobile apps and the tiqr server-side reference implementation.

SURFnet is now at a stage where it wants to start deploying tiqr, both within SURFnet, as well as within its constituency. Before this can be done, SURFnet felt that it would be necessary to have an independent party perform a security audit on the tiqr technology. The goals set for this security audit were to:

- Assess the tiqr architecture and determine what attack surfaces exist and how these can be protected;
- Perform code audits of the tiqr apps and server-side reference implementation;
- Assess the security of tiqr in a live environment

The Dutch security company Madison Gurkha[1] was selected to perform this audit. The actual audit was performed in May of 2011, after which SURFnet and Egeniq made alterations to the tiqr apps and server-side reference implementation to deal with a number of vulnerabilities that were identified in the security audit.

## 2.2 Methodology

Madison Gurkha performed a so-called *crystal box* security assessment. This means that they had full access to all information pertaining to the system in advance. By request from Madison Gurkha a detailed discussion of the security testing techniques used by Madison Gurkha to perform the audit is not included in this document. This information is available in the full audit report that is available on request (see §1.1).

Every vulnerability identified has been classified according to two criteria:

- *Likelihood* (the probability of exploitation of a vulnerability); the likelihood is classified in one of three categories:
    - *Low* – exploitation requires specific actions from a knowledgeable attacker
    - *Medium* – exploitation requires medium skill and knowledge such as writing simple programs or scripts

[1] http://www.madison-gurkha.com/en/index.php

  - *High* – exploitation requires little skill, minimal effort and the use of readily available tools

- *Consequences* (the effect of exploitation of a vulnerability); the consequences are classified as lying in one of three categories:
  - *Low* – there are no useful or limited consequences financial or otherwise (e.g. leaking version information)
  - *Medium* – there are limited and/or quantifiable financial consequences or reputation damage (e.g. use of weak cryptography)
  - *High* – there are substantial financial consequences or there is substantial reputation damage (e.g. access to sensitive data without proper authorisation)

Based on these two classification criteria, every vulnerability that was identified is placed in a risk category (**High**, **Medium** or **Low**) according to the following table:

| Likelihood \ Consequences | Low | Medium | High |
|---|---|---|---|
| Low | **Low** | **Low** | **Medium** |
| Medium | **Low** | **Medium** | **High** |
| High | **Medium** | **High** | **High** |

**Table 1 - Classification of vulnerabilities**

Every issue described in chapter 3 has been categorised according to this methodology.

# 3 Issues, changes and mitigation

## 3.1 Introduction

This chapter lists the issues that were identified during the audit. The issues are listed according to their risk category, from high to low. In addition to identifying a number of risks, the auditors also made some remarks suggesting a number of improvements. These are also listed at the end of this chapter.

For each issue, we include a description of the mitigation that was implemented to deal with the issue.

Finally, the risk mitigation solutions described in this chapter were presented to Madison Gurkha and used as a basis for a re-audit of tiqr that was performed in September of 2011. The feedback provided by the auditors on the actions that were taken and the effect this had in terms of security is also included for each issue that was identified.

## 3.2 High risk vulnerabilities

### 3.2.1 HR-1 Buffer of size 0

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS tiqr app |
| Description: | It is possible to craft input data (by specifying and unsupported hash algorithm) for the enrolment process that can trigger creation of a buffer of 0 bytes in the iOS tiqr app |
| Consequences: | An attacker can cause a buffer overflow in this buffer, which can crash the application, may allow execution of arbitrary code or unauthorised access to sensitive data |

**Mitigation**

The OCRA implementation has been changed to only accept supported algorithms. If an unsupported algorithm is specified as input, the application will fall back to the default algorithm (HMAC-SHA1) to prevent an uninitialised variable leading to the crash specified in the security audit.

Note that this may lead to an unusable enrolment if an invalid algorithm is specified (this is, however, fail safe).

This issue has been resolved in tiqr 1.1.1 for iOS (available from the App Store).

**Auditor remarks**

*A default algorithm is chosen. The algorithm can then be overwritten if a valid algorithm is provided. This effectively resolves the issue as indicated.*

### 3.2.2 HR-2 Phishing during enrolment

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS and Android tiqr app |
| Description: | The enrolment URL that is used to send the users new credentials to is not displayed to the user during the enrolment process |
| Consequences: | An attacker can execute a man-in-the-middle attack during enrolment without the user detecting this attack and can illicitly gain access to the user's secret key |

**Mitigation**

The iOS and Android tiqr app have been modified to show the fully qualified domain name of the enrolment server. This allows the user to verify that enrolment data is getting sent to a trusted server (by comparing the domain name to the URL in the browser).

Note 1: auto-enrolment such as is performed in the demonstration setup is atypical; it is assumed that enrolment is a strictly regulated process according to organisational policy in a production setup.

Note 2: part of the solution to this problem is user education (this is true for all phishing problems).

This issue has been resolved in tiqr 1.1.1 for iOS (available from the App Store) and in tiqr 1.1.0 for Android (available from the Android Market).

**Auditor remarks**

*[Given the explanation provided above and our testing experience] we consider this issue to be resolved.*

### 3.2.3 HR-3 Phishing during authentication

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS and Android tiqr app |
| Description: | The site to which the user is authenticating is not displayed to the user during the authentication process |
| Consequences: | An attacker can execute a man-in-the-middle attack during authentication and thus gain access to the user's account |

**Mitigation**

The iOS and Android tiqr app already show the fully qualified domain name of the authentication server to allow the user to verify that enrolment data is getting sent to a trusted server (by comparing the domain name to the URL in the browser). This is therefore a matter of user education (as is true for all phishing problems).

Note: tiqr is no more or less vulnerable to phishing then any other two-factor authentication system based on challenge/response (such as e.g. the OTP tokens currently in use by banks or for instance SMS authentication). It was not a specific design goal for tiqr to prevent phishing.

**Auditor remarks**

*The FQDN [fully qualified domain name] is shown during authentication. It is up to the user to verify this URL. We consider this issue to be resolved.*

### 3.2.4 HR-4 Unauthorised access to server-side data

**Description of the issue**

| | |
|---|---|
| Affected components: | Server-side reference implementation (demo server) |
| Description: | There is insufficient input validation on the data supplied during enrolment |
| Consequences: | An attacker can create files in arbitrary locations on the server file system with the privileges of the application server user |

**Mitigation**

Input validation on the user ID has been implemented. Furthermore, user IDs are now escaped before they are displayed preventing client side rendering of embedded HTML or execution of embedded script code.

This issue has been resolved in the *trunk* revision of the reference implementation in the tiqr Subversion repository and on the tiqr demo server.

**Auditor remarks**

*The user ID is now validated and dangerous characters are no longer allowed. We consider this issue to be resolved.*

#### 3.2.5 HR-5 Cross-site scripting

**Description of the issue**

| | |
|---|---|
| Affected components: | Server-side reference implementation (demo server) |
| Description: | The reference implementation contains a cross-site scripting vulnerability due to insufficient input validation in the handling of enrolment data |
| Consequences: | An attacker can enroll a user with cross-site scripting code embedded in the user information (e.g. the user's Display Name) and then use this user account in concurrence with the file metadata.php to execute the embedded scripting code in the context of the browser user, thus e.g. gaining access to a user's session cookies |

**Mitigation**

The display name user attribute is now escaped in all output to mitigate this vulnerability.

This issue has been resolved in the *trunk* revision of the reference implementation in the tiqr Subversion repository and on the tiqr demo server.

**Auditor remarks**

*Initial re-audit of this vulnerability showed that it had not been resolved. SURFnet was notified and immediately took action to resolve the issue. We consider this issue to be resolved.*

## 3.3 Medium risk vulnerabilities

#### 3.3.1 MR-1 Trust store

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS and Android tiqr app |
| Description: | The default trust store for CA certificates is used to determine whether or not the TLS connection to enrolment and authentication services is trusted |
| Consequences: | Users cannot control which CAs are trusted; conversely any server with a CA-issued certificate can act as enrolment or authentication server. From a security perspective it is desirable to be able to control this in high-security environments |

**Mitigation**

This issue (using the default trust store) is currently not seen as a major issue since almost all mobile applications do this. SURFnet sees this feature as an additional security measure that can be taken in high-security environments. Because tiqr is open source, it is easy for those who feel they need this feature to implement it for their environment.

**Auditor remarks**

*The issue has – as indicated by SURFnet – not been resolved.*

### 3.3.2 MR-2 TLS cipher selection

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS and Android tiqr app |
| Description: | A default TLS connection object is used to set up the connection to an enrolment or authentication server, which may mean that the connection supports weak cipher (this depends on the implementation of the TLS object in the mobile OS) |
| Consequences: | If the enrolment or authentication server also allows weak ciphers an attacker can perform a man-in-the-middle attack and force use of a weak cipher |

**Mitigation**

This issue can be effectively mitigated by correctly configuring the authentication server to only accept strong cryptographic algorithms.

**Auditor remarks**

*As stated in the resolution, the issue can be mitigated by configuring the server to only accept strong secure ciphers. For applications that do not require very high security, this mitigation factor will be sufficient.*

### 3.3.3 MR-3 Input validation

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS and Android tiqr app, server-side reference implementation (demo server) |
| Description: | Not all input is validated before processing |
| Consequences: | No immediate vulnerabilities were identified, however, the lack of input validation may leave open as yet unforeseen opportunities to attack the system |

**Mitigation**

All data is escaped properly to prevent e.g. malicious data containing script commands or SQL statements from executing.

This issue has been resolved in tiqr 1.1.1 for iOS (available from the App Store) and in tiqr 1.1.0 for Android (available from the Android Market).

**Auditor remarks**

*The solution does not address the issue reported in Medium Risk 3 (of the original report), but rather fixes possible risks associated with the issue. The main problem is the lack of input validation for the applications. Ideally, input validation takes place as early as possible, before it is in any way passed on or stored in the application. In the iOS code, we see that the metadata, sent by the server as a JSON object is only partially validated. We recommend that extra validation is added.*

*The issue is not resolved, although we have not found any places where this can be exploited in the applications and the server libraries.*

### 3.3.4 MR-4 Buffer overflow

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS tiqr app |
| Description: | Requesting an OCRA suite with an output of more than 10 digits can trigger an array overflow in the code because the input data is not validated sufficiently |
| Consequences: | The application may crash causing a denial of service |

**Mitigation**

The code now checks that the number of returned OCRA digits as specified in the OCRA suite does not exceed 10 digits. This problem was also present in the OCRA reference implementation and a fix has been contributed back to the RFC authors.

This issue has been resolved in tiqr 1.1.1 for iOS (available from the App Store) and in tiqr 1.1.0 for Android (available from the Android Market).

**Auditor remarks**

*The number of digits is effectively checked and an error is returned if it is too large. For iOS, the issue is effectively resolved.*

*For Android, we cannot find any check in the source code. There was no buffer overflow present, but rather an ArrayOutOfBoundsException that was thrown, crashing the application, making the risk less high. We check the different points where the OCRA code is used and see that the ArrayOutOfBound- sException is caught and replaced by an error. The application will not crash due to the invalid output. We consider the issue to be resolved.*

### 3.3.5 MR-5 Return URL abuse

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS and Android tiqr app |
| Description: | No input validation is performed on the optional return URL included in an authentication challenge |
| Consequences: | It is – for example – possible to trigger the handset to dial a phone number on iOS by setting a return URL that starts with 'tel:' |

**Mitigation**

The code has been modified to only accept return URLs starting with http:// or https://.

This issue has been resolved in tiqr 1.1.1 for iOS (available from the App Store) and in tiqr 1.1.0 for Android (available from the Android Market).

**Auditor remarks**

*[Inspection of the] code shows that the value is now correctly validated on the iOS and Android platforms.*

*We consider this issue to be resolved.*

### 3.3.6 MR-6 Access to credentials on compromised handset

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS and Android tiqr app |
| Description: | An attacker can install a malicious version of the tiqr app on a compromised handset (e.g. jailbroken iPhone) and gain illicit access to user credentials |
| Consequences: | A users credentials can become compromised |

**Mitigation**

This issue cannot be resolved other than through user education. Any solution that relies on applications running on the phone (such as – for instance – many one-time password apps currently available from traditional OTP vendors) is vulnerable to this issue.

The protection of application secrets also differs from platform to platform. On iOS, protection is quite strong since the key chain that contains the tiqr secrets is application-specific for the tiqr app and the encryption of key-chain data is tied to the application vendor certificate (during development it was discovered that a

tiqr app signed with a different – e.g. development – certificate cannot decrypt the data stored in the application's key chain). On Android, such protection is not present.

As long as an attacker does not have access to the user's PIN code, the user secret is also protected. It is encrypted in unstructured form using the PIN code as a basis. By default, servers are configured to block user accounts after 3 failed authentication attempts, effectively stopping a brute force attack.

**Auditor remarks**

*As noted, the issue cannot be resolved other than by introducing new hardware. If the smartphone is hacked, an attacker will be able to install a keylogger to intercept the PIN and thus also gain access to the PIN. We consider the issue to be unsolved.*

### 3.3.7 MR-7 Denial-of-service on server

**Description of the issue**

| | |
|---|---|
| Affected components: | Server-side reference implementation (demo server) |
| Description: | After three failed login attempts an account is blocked |
| Consequences: | An attacker who knows a user's ID can block a user's account and cause a denial-of-service |

**Mitigation**

For security reasons it is necessary to block accounts after a pre-configured number of failed login attempts (see also MR-6 §3.3.6). The tiqr protocol has been extended to also accept temporary blocking of accounts (e.g. blocking an account for 30 minutes after a certain number of failed attempts to throttle and detect brute force attacks). It is, however, not possible to completely stop this form of denial-of-service attack.

Note: any current OTP or password-based system suffers from this problem in some form or other; it is not unique to tiqr.

**Auditor remarks**

*As noted, the issue cannot be resolved. Adding a temporary blocking of accounts will mitigate the issue, as most attacks will switch to new targets, after which the account will automatically be unlocked.*

*The following recommendation is made to users wanting to implement tiqr in their application [referral to section 4.2 of this document].*

*Based on the mitigation and the recommendation, we consider the remaining issue to be either a low risk or, depending on the actual server configuration, to be resolved.*

## 3.4 Low risk vulnerabilities

### 3.4.1 LR-1 No OCSP or CRL support on mobile platforms

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS and Android tiqr app |
| Description: | There is no OCSP or CRL check on the mobile platform when setting up a TLS connection |
| Consequences: | Revoked server certificates are not detected when setting up a TLS connection |

**Mitigation**

This is a platform dependent issue that cannot be resolved easily by changing the tiqr application. The most probable reason why these features are not present on

mobile devices is that they require data exchanges (which may be charged for and consume battery power). The risk posed by not implementing these features on a platform is deemed to be low.

**Auditor remarks**

*The (small) risk remains present. We do agree that it is more of a platform dependent issue. New updates might even add the functionality automatically.*

### 3.4.2 LR-2 Uncaught exception

**Description of the issue**

| | |
|---|---|
| Affected components: | Android tiqr app |
| Description: | This is the Android equivalent of HR-1 (§3.2.1); on Android, however, the buffer of size 0 will lead to an uncaught ArrayIndexOutOfBoundsException in the Java code |
| Consequences: | The application will crash with an error message |

**Mitigation**

The tiqr app now checks if the challenge that was sent is of acceptable length; if it is not, the app will report an error message to the user stating that the app is incompatible with the authentication server.

This issue has been resolved in tiqr 1.1.0 for Android (available from the Android Market).

**Auditor remarks**

*[Inspection of the code] shows that the input is now validated. The length of the question may not be larger than 254 bytes. We consider the issue to be resolved.*

### 3.4.3 LR-3 Error message shows internal path names

**Description of the issue**

| | |
|---|---|
| Affected components: | Server-side reference implementation (demo server) |
| Description: | If an error occurs on the server side, the resulting error page shows internal pathnames |
| Consequences: | Information about the internal file system organisation of the server is revealed |

**Mitigation**

The tiqr demo server has been configured in debug mode (this is a SimpleSAMLphp feature). The result of this is that stack traces are printed in case of an error occurring; this is not a tiqr feature but rather a configuration decision for the demo server and thus does not require mitigation.

**Auditor remarks**

*This configuration is intentional for the demo server. As it is a server configuration for a demo setup, the finding is not relevant for the general security of the tiqr application. The finding is not taken into account for the assessments.*

### 3.4.4 LR-4 Server reveals whether a user exists (reference implementation)

**Description of the issue**

| | |
|---|---|
| Affected components: | Server-side reference implementation (demo server) |
| Description: | The server gives information about whether or not a username exists |
| Consequences: | This makes a brute-force denial-of-service attack possible by leveraging MR-7 (§3.3.7) |

**Mitigation**

This is specific to the enrolment scenario used in the demo setup; it is perfectly feasible to design an enrolment process that does not have this drawback.

No changes have been made to mitigate this.

**Auditor remarks**

*This configuration is intentional for the demo server. As it is a server configuration for a demo setup, the finding is not relevant for the general security of the tiqr application. The finding is not taken into account for the assessments.*

*Recommendations are made to administrators wanting to setup tiqr for their application [reference to section 4.2 of this document].*

*We consider the issue to be resolved.*

### 3.4.5 LR-5 Server allows dynamic module loading

**Description of the issue**

| | |
|---|---|
| Affected components: | SimpleSAMLphp (demo server) |
| Description: | The software used to run the demo server (SimpleSAMLphp) allows dynamic loading of server plugin modules |
| Consequences: | An attacker may be able to access unnecessary plugin modules |

**Mitigation**

This is a matter of configuring SimpleSAMLphp correctly and as such falls outside the scope of the tiqr design and implementation.

No changes have been made to mitigate this.

**Auditor remarks**

*This is a server configuration for a demo setup; the finding is not relevant for the general security of the tiqr application. The finding is not taken into account for the assessments.*

## 3.5 Remarks

### 3.5.1 RM-1 tiqr is not full 2-factor authentication

**Description of the issue**

| | |
|---|---|
| Affected components: | Design |
| Description: | The mobile device is not a dedicated hardware device and user credentials may be cloned onto secondary devices (e.g. using iOS backup and restore) |
| Consequences: | None |

**Mitigation**

It is a matter of discussion what is and is not full 2-factor authentication. For tiqr, the user must at least have a device (something he/she has) and know a PIN code (something he/she knows). This complies with the classic definition of 2-factor authentication.

The biggest difference with what could be called a full 2-factor authentication solution is that it is easier to make a copy of the token (e.g. restoring a backup of an iPhone onto a new device copies the user credentials and putting the memory card from an Android phone in a new device migrates the identity). Nevertheless, to use a tiqr identity a user will still need to both have the credentials belonging to an identity as well as know the PIN code.

Note that this discussion also applies to other app-based 2-factor authentication solutions such as the OTP apps available from many OTP token vendors.

**Auditor remarks**

*We have no further comments on this issue.*

### 3.5.2 RM-2 Inconsistent states are possible

**Description of the issue**

| | |
|---|---|
| Affected components: | iOS and Android tiqr app |
| Description: | Inconsistent states between server and client are possible |
| Consequences: | An enrolment failure can cause a state inconsistency between the client and the server. This makes a partially enrolled account unusable. |

**Mitigation**

Because of the nature of the information exchange between tiqr client and server it is not possible to give a 100% guarantee that a transaction has been successful and thus that both client and server have the same state. This is, however, not a security issue since the worst thing that could happen is that a user needs to re-enrol an identity.

**Auditor remarks**

*We have no further comments on this issue.*

### 3.5.3 RM-3 SQL statements used in unsafe manner (Android app)

**Description of the issue**

| | |
|---|---|
| Affected components: | Android tiqr app |
| Description: | Code that performs an SQL query is not a prepared statement |
| Consequences: | In theory, this could lead to a vulnerability to SQL injection (this is, however, not the case here since the variable used in the SQL statement is an integer value) |

**Mitigation**

The code has been changed such that the theoretical SQL injection is no longer possible.

This issue has been resolved in tiqr 1.1.0 for Android (available from the Android Market).

**Auditor remarks**

*We can no longer find the use of unsafe SQL in the source code. We consider the issue to be resolved.*

# 4 Conclusions and recommendations

## 4.1 Conclusions

The security audit has revealed several issues, which were dealt with successfully by changing the tiqr app on both iOS as well as Android. The audit also shows that the design premises behind tiqr are sound.

What is important to note is that tiqr does not protect users against all security issues associated with authentication. Just like almost all OTP-based authentication mechanisms, tiqr is vulnerable to phishing attacks on the authentication channel. We have, however, taken precautions that should enable users to more successfully deal with such problems because we can leverage the richer user experience a smart phone can offer.

It should also be noted that – like other app-based OTP solutions – tiqr does not implement full two-factor authentication but is rather a hybrid solution.

Finally, we believe that tiqr offers significant benefits over the traditional username/password paradigm and also has benefits in terms of user experience over more traditional OTP solutions.

We include the conclusions drawn by the auditors based on the re-audit below:

*"During the reassessment, we have identified only two relevant medium risks, and one relevant low risk for tiqr. In some places of the client applications and server libraries, we have found insufficient input validation. Although this cannot be abused by an attacker, we consider it good practice to validate all input before processing it. A future change in the code might introduce code where invalid input does cause a problem.*

*The second medium risk is not specific for tiqr, but we consider it important enough and want to make sure that people are aware of the issue. If the smartphone is compromised without the user's knowledge, then an attacker may be able to obtain the credentials. Note that this is equally true for username/password authentication and client certificates stored on the device.*

*Based on the findings made during the re-audit, we consider the security of the tiqr applications and server libraries to be more than sufficient."*

## 4.2 Recommendations

We would like to recommend addressing the following things when planning a tiqr deployment or considering designing your own tiqr server implementation:

- Ensure that a sound enrolment process is established; the strength of any identity – including a tiqr identity – heavily relies on this. For example: if users can auto-enrol through a web portal to which they authenticate using a username and password means that the resulting tiqr identity has the same level-of-assurance that the username and password offer.
- Educate users about the risks of phishing and how this can be prevented by enacting simple checks that tiqr facilitates (by showing domain names of authentication servers to which the user is authenticating).
- Carefully consider naming conventions for your tiqr server infrastructure; it is counterintuitive for users – and imposes a risk of phishing – if the authentication server's domain name differs from the URL through which the user access the authentication site.
- If you implement your own tiqr server components based on the reference implementation make sure that you store the user credentials securely.

- Carefully think about how your server will deal with failed authentication attempts (e.g. by blocking accounts and by monitoring anomalous behaviour indicating brute force attacks).