Quipux Vulnerability Report

Ola Bini, Principal Security Analyst

2025-11-01

Contents

1 Introduction

This technical report describes a number of vulnerabilities found in Quipux by the team at Seguridad Digital EC. These vulnerabilities were identified by analysis of the publicly available source code. While certain countermeasures exist, they are not sufficient to protect against a number of problematic conditions. The team practices responsible disclosure and have given the provider sufficient time to fix the problems identified before publication.

This report describes 24 cases of SQL injection. These vulnerabilities can only be exploited after authentication, but due to the open nature of the injections, any one of them compromise the full database. This report also describes one case of reflected cross-site scripting (XSS) which can be used to compromise user accounts. Finally, one case of data disclosure was found.

The vulnerabilities identified have been assigned CVE-2025-55341 for the XSS vulnerability, CVE-2025-55342 for the information disclosure, and CVE-2025-55343 for the 24 SQL injections. At the time of publication, these are still reserved.

Seguridad Digital EC follow the standards for responsible disclosure used by Google Project Zero. This implies giving the provider - the Ministry of Telecommunications and Information Society - 90 days counting from the time of disclosure to correct any problems identified. At the time of publication of this report, several of the vulnerabilities have been corrected, while others are still active.

The original source code analyzed was published at https://minka.gob.ec/quipux-comunitario/quipux-comunitario but during the process of correcting the issues, this repository has been deleted by the provider. The new source code can be found at https://minka.gob.ec/mintel/ge/quipux/quipuxcomunitario. However, for reference, the original source code analyzed can now be found at https://github.com/seguridaddigitalec/qu old.

The team at Seguridad Digital EC would like to officially thank Fluid Attacks for the help in reporting the CVE:s in question. The team would also formally thank the team at the Ministry of Telecommunications for their receptiveness and willingness to receive our report and fixing the problems identified.

2 What is Quipux

Quipux is a document management system for the public sector developed in Ecuador, based on an older code base from Colombia. The system is free software and publicly available for review. A number of public institutions in Ecuador run versions of Quipux, including the central government and many universities. The system is maintained by the Ministry of Telecommunications and Information Society. It is not completely clear exactly which versions of Quipux are deployed in different settings. The publicly available source code has not been updated for some time, and it seems that the installed versions have functionality that is not visible in the source code. For this reason, it is possible that development have continued in private and that the newest source code is not publicly available (even though that is contrary to the license of the software, and also to the laws and regulations of Ecuador).

Conservative estimations lead us to believe that there exists at least 20 but more likely many more installations in the country. It is also well known that the system is in daily use by almost all functionaries in the public sector, making it a system that has hundreds of thousands of daily users. Even though it's unclear what code is running in some of these institutions, it seems likely that most of the vulnerabilities found are conserved in these installations.

Since the official record keeping and digital signature systems run inside of Quipux, it can't be overstated how important this system is for the management of the country of Ecuador. Any kind of vulnerability should be considered extremely serious.

3 Vulnerabilities

All of the vulnerabilities documented in this report were identified in commit `e1774acd75e4c538413a137304cce` (which is the latest in the old repository). However, all vulnerabilities have been identified in earlier versions going back several years. All vulnerabilities have been fixed in the newest version of the code base.

3.1 SQL Injection

24 instances of SQL Injection were found in the code base. These are tracked as CVE-2025-55343. These map to CWE-89 and CWE-74. The root cause for SQL Injection happens when a database query is made in such a way that data controlled by an end user is composed with the query in such a way that its semantics change. This can lead to a large range of impacts, depending on what other protections exist. In the most extreme case, the full content of the database can be extracted, and any content can also be modified. In certain cases, complete remote code execution can also be achieved, giving an attacker full control of the server in question. In the current case, all these possibilities are likely achievable. Evaluating the situation using CVSS 3.1 gives the following parameters: `AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H`, and a score of 9.9. The main reason why a full 10 is not achieved is because all 24 vulnerabilities are protected by the need for authentication. In other words, a user has to authenticate before being able to execute an attack. But any authenticated user can achieve the attacks in question.

The Quipux system does have functionality to protect against SQL Injection. This is done using the function `limpiar_sql()`. The main impediment imposed by this function is the inability to use a single quote character (’). While this is sufficient when user-controlled content is put inside of a string element in the database query, it does not have much effect in the 24 instances identified where an injection is possible, since the injection happens outside of a string element.

3.1.1 SQL Injection in search functionality (sqli.1)

This vulnerability can be found in `busqueda/busqueda.php` on line 86. It happens after the check for authentication. The injection is on the POST parameter `txt_depe_codi`, and is located at the

end of the SQL query.

3.1.2 SQL Injection in search functionality 2 (sqli.2)

This vulnerability can be found in `busqueda/busqueda.php` on line 92. It happens after the check for authentication. The injection is on the POST parameter `txt_usua_codi`, and is located at the end of the SQL query.

3.1.3 SQL Injection in attachments (sqli.3)

This vulnerability can be found in `anexos_lista.php` on line 66. It happens after the check for authentication. The injection is on the GET parameter `radi_temp`, and is not located at the end of the SQL query.

3.1.4 SQL Injection in administration forms (sqli.4)

This vulnerability can be found in `Administracion/listas/formArea_ajax.php` on line 25. It happens after the check for authentication. The injection is on the GET parameter `codDepe`, and is located at the end of the SQL query.

3.1.5 SQL Injection in administration forms 2 (sqli.5)

This vulnerability can be found in `Administracion/listas/formDepeHijo_ajax.php` on line 23. It happens after the check for authentication. The injection is on the GET parameter `codDepe`, and is not located at the end of the SQL query.

3.1.6 SQL Injection in administration forms 3 (sqli.6)

This vulnerability can be found in `Administracion/listas/formDepePadre_ajax.php` on line 23. It happens after the check for authentication. The injection is on the GET parameter `codInst`, and is not located at the end of the SQL query.

3.1.7 SQL Injection in associated documents (sqli.7)

This vulnerability can be found in `asociar_documentos/asociar_borrar_referencia.php` on line 17. It happens after the check for authentication. The injection is on the POST parameter `radi_nume`, and is located at the end of the SQL update instruction.

3.1.8 SQL Injection in associated documents 2 (sqli.8)

This vulnerability can be found in `asociar_documentos/asociar_documento_buscar_query.php` on line 74. It happens after the check for authentication. The injection is on the GET parameter `radi_nume`.

3.1.9 SQL Injection in associated documents 3 (sqli.9)

This vulnerability can be found in `asociar_documentos/asociar_documento_grabar.php` on line 53. It happens after the check for authentication. The injection is on the POST parameter `txt_radi_nume`.

3.1.10 SQL Injection in associated documents 4 (sqli.10)

This vulnerability can be found in `asociar_documentos/asociar_documento.php` on line 33. It happens after the check for authentication. The injection is on the GET parameter `radi_nume`.

3.1.11 SQL Injection in location functionality (sqli.11)

This vulnerability can be found in `radicacion/buscar_usuario.php`. It happens after the check for authentication. The injection is on the POST parameter `buscar_tipo`.

3.1.12 SQL Injection in location functionality 2 (sqli.12)

This vulnerability can be found in `radicacion/formArea_ajax.php`. It happens after the check for authentication. The injection is on the GET parameter `codDepe`.

3.1.13 SQL Injection in location functionality 3 (sqli.13)

This vulnerability can be found in `radicacion/formDepeHijo_ajax.php`. It happens after the check for authentication. The injection is on the GET parameter `codDepe`.

3.1.14 SQL Injection in location functionality 4 (sqli.14)

This vulnerability can be found in `radicacion/formDepePadre_ajax.php`. It happens after the check for authentication. The injection is on the GET parameter `codInst`.

3.1.15 SQL Injection in location functionality 5 (sqli.15)

This vulnerability can be found in `radicacion/ver_datos_usuario.php`. It happens after the check for authentication. The injection is on the GET parameter `destinatario`.

3.1.16 SQL Injection in reports (sqli.16)

This vulnerability can be found in `reportes/reporte_TraspasoDocFisico.php`. It happens after the check for authentication. The injection is on the GET parameter `verrad`.

3.1.17 SQL Injection in transactions (sqli.17)

This vulnerability can be found in `tx/datos_imprimir_sobre.php`. It happens after the check for authentication. The injection is on the GET parameter `txt_usua_codi`.

3.1.18 SQL Injection in transactions 2 (sqli.18)

This vulnerability can be found in `tx/datos_imprimir_sobre.php`. It happens after the check for authentication. The injection is on the GET parameter `nume_radi_temp`.

3.1.19 SQL Injection in transactions 3 (sqli.19)

This vulnerability can be found in `tx/datos_imprimir_sobre.php`. It happens after the check for authentication. The injection is on the GET parameter `nume_radi_temp`.

3.1.20 SQL Injection in transactions 4 (sqli.20)

This vulnerability can be found in `tx/revertir_firma_digital_grabar.php`. It happens after the check for authentication. The injection is on the GET parameter `txt_radi_nume`.

3.1.21 SQL Injection in transactions 5 (sqli.21)

This vulnerability can be found in `tx/tx_borrar_opcion_imp.php`. It happens after the check for authentication. The injection is on the GET parameter `codigo_opc`.

3.1.22 SQL Injection in transactions 6 (sqli.22)

This vulnerability can be found in `tx/tx_realizar_tx.php`. It happens after the check for authentication. The injection is on the GET parameter `txt_radicados`.

3.1.23 SQL Injection in transactions 7 (sqli.23)

This vulnerability can be found in `tx/tx_seguridad_documentos.php`. It happens after the check for authentication. The injection is on the GET parameter `txt_radicados`.

3.1.24 SQL Injection in file upload (sqli.24)

This vulnerability can be found in `uploadFiles/cargar_doc_digitalizado_paginador.php`. It happens after the check for authentication. The injection is on the GET parameter `txt_depe_codi`.

3.2 Cross-Site Scripting (XSS)

The file `anexos/anexos_nuevo.php` contains a Cross-Site Scripting (XSS) vulnerability. This can be found on line 23, where the POST parameter `asocImgRad` is directly inserted into the printed HTML, without any cleaning or verification of data. This page can be accessed without authentication. This vulnerability is tracked as CVE-2025-55341 and is classified as CWE-79. Using CVSS 3.1, its parameters are `AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N`. Based on all available information, it gets a score of 6.5.

This vulnerability can be used by an attacker to achieve a range of different actions, including the theft of cookies and other private material. It can also be used to automatically execute actions impersonating the user. The fact that it is a POST parameter makes these attacks more complicated, but the fact that the application does not have any type of protection such as CSRF tokens makes this vulnerability more serious and a threat to the whole application.

3.3 Data disclosure

The file `Administracion/usuarios/cambiar_password_olvido_validar.php` contains a data disclosure vulnerability. It can be accessed without authentication. When given a POST parameter `txt_login` it will return information about the account with the username matching the given data. The information revealed include the cedula (the Ecuadorean identity number) of the user

in question, and the type of user. Making things worse, the `txt_login` parameter is executing a search using the SQL "LIkE" operator, that allows the use of subsets of the username. This can be used to easy enumerate the full database of users in the database.

This vulnerability is tracked as CVE-2025-55342 and is classified as CWE-200. Using CVSS 3.1 its parameters are `AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N`. Based on all available information, it gets a score of 5.3.

4 Recommendations

It is generally not considered appropriate to try to clean user generated input in order to avoid SQL Injection. Instead, it is always recommended to use so called parameterized queries, which makes injection vulnerabilities impossible. Most programming languages have complete support for this feature, and PHP is no exception.

In order to avoid Cross-Site Scripting (XSS) vulnerabilities it is recommended to always clean user provided data before printing it as part of HTML pages. It is also recommended to use other features, such as Content Security Policies to avoid executing of arbitrary JavaScript in regular pages. This requires proper separation between scripts and HTML pages. Finally, it is important to apply CSRF tokens and other counter measures to reduce the impact of potential XSS vulnerabilities.

In general, it is a good idea to properly define APIs for accessing account data. These APIs should have correct authentication and authorization defined, and should never expose more information than the user should have access to.

In order to reduce the impact of these vulnerabilities, it is possible to use Web Application Firewalls or Next Generation Firewalls, until it is possible to update to the latest version with fixes applied.

5 Timeline

5.1 Week of April 28, 2025

Analysis of Quipux source code done by team at Seguridad Digital EC. All vulnerabilities discovered and verified.

5.2 May 3rd, 2025

Initial report sent to EcuCERT. Request for 3 CVEs sent to Mitre.

5.3 May 9th, 2025

Initial report sent to servicios@gobiernoelectronico.gob.ec. An automated response given that created the ticket “2025-14002”.

5.4 May 30th, 2025

Physical letter to the Minister of Telecommunications entered at the Ministry.

5.5 June 12th, 2025

Answer received from Edgar Roberto Acosta Andrade. Response sent to Cristian Cartuche by Ola Bini.

5.6 June 16th, 2025

Response received from Cristian Cartuche, proposing meeting on Wednesday 18 of June at 15:00.

5.7 June 17th, 2025

Response to Cristian Cartuche sent, proposing other dates and times for meeting.

5.8 June 18th, 2025

Response from Cristian Cartuche received, proposing Friday 20 of June at 15:00. Response sent to Cristian, confirming. Response received from Cristian, confirming.

5.9 June 20th, 2025

Full notification and details of vulnerabilities reported to the Ministry.

5.10 Aug 12th, 2025

3 CVEs reserved.

5.11 Sep 4th, 2025

New codebase with partial fixes uploaded to new source code repository.

5.12 Sep 10th, 2025

Communication with the provider about the coming disclosure and notification that fixes are not complete.

5.13 Sep 16th, 2025

Communication from the provider received, asking for a meeting related to timeline of disclosure.

5.14 Sep 17th, 2025

Response to provider, proposing a meeting time and reminding provider about the 90 day deadline.

5.15 Sep 18th, 2025

Meeting with subsecretary and other functionaries. Decision made that Seguridad Digital EC will work with developers from the ministry to fix remaining vulnerabilities. Deadline for publication extended. November 1, 2025 is the new date of publication.

5.16 Oct 22nd, 2025

A technical meeting was held with the Ministry to review progress and provide support in implementing the remaining fixes.

5.17 Nov 1st, 2025

This report published.

6 References

- https://www.ecucert.gob.ec/
- https://minka.gob.ec/mintel/ge/quipux/quipuxcomunitario
- https://minka.gob.ec/quipux-comunitario/quipux-comunitario
- https://web.gestiondocumental.gob.ec
- https://github.com/seguridaddigitalec/quipux-old