Security of user authentication in payment systems in the agricultural value chain

N Polyakov1, F Tishkin1, L Cherkesova1, O Safaryan1

1Don State Technical University, pl. Gagarina, 1, 344002, Rostov–on–Don, Russia

E-mail: safari_2006@mail.ru

Abstract. In the offered article, the threats of informational safety, arising in the modern world, directed on the bank and commercial payment systems using in all world and in Russia in particular are considered. The circuits of operation of the most popular Russian banking and commercial payment systems using electronic money (e–cash) and their possible vulnerability caused by increase of monetary calculations through the Internet are described. Not all bank devices support biometric scanning; therefore, there is question on how the huge amount of bank payment systems users can protect their personal data from not authorized operations (hackers breaking)? For support of payments safety, the paper offers using the additional complicated authentication – the application of simplified cryptography token which is looking like program application, and installed on USB – device. It will allow amplifying the safety at authentication stage as the program application generating of one – time keywords and checking (controlling) of user’s authenticity with the help of white list, is capable to eliminate all existing vulnerabilities and gaps in the informational protection of banking and commercial payment systems.

1. Introduction

Payment systems are the units that provide convenience to bank customers when conducting cash transactions. In Russia today, the most famous are the Visa International Service Association, MasterCard International (WorldWide or Incorporated) and the NPCS MIR. Visa and MasterCard are the leading international payment systems and are serviced throughout the world; MIR is the Russian national payment system that is mandatory for all employees of state organizations [1].

In recent years, payments through the Internet are gaining increasing popularity. They are executed with the support of websites or client applications. Banks develop personal secure authentication systems. For example, the latest development of Sberbank is Sberbank ID. But not all devices support biometric scanning and the question arises, and how to protect a huge mass of users. To ensure the security of payments, we recommend the introduction of additional sophisticated authentication - the use of a simplified cryptographic token, which will be in the form of an application and installed on a USB device.

2. The main goal

The main task of our work is to check the vulnerabilities of the authentication stage in an electronic wallet and solve the problems found by creating a cryptographic token created on Python in the Pinentry2 application. To accomplish this goal, we need to solve a number of tasks: consider probable attacks on the user while logging in to the website of the payment system, create an auxiliary check using a randomly generated one-time password on external media, and create a white list of IP addresses.
3. Electronic payment systems

Electronic payment systems are services where it is possible to create a personal account with an electronic wallet attached to it that tracks the movement of electronic money. They make it possible to carry out the same monetary transactions, in fact, as in the case of a regular card. The most popular payment services in Russia [2]:

1. Yandex.Money;
2. WebMoney;
3. QIWI;
4. PayPal.

The bulk of the systems are considered not anonymous or partially anonymous. Payment for goods through these systems offers virtually any online store.

To ensure the security of their customers, payment systems use all sorts of ways.

For example, in a digital cash model, the security is provided by the persistence of cryptographic protocols used in the manufacture (emission) of digital money and regulating their turnover.

Like cash notes, they contain information about the nominal value, issuer, series, number, as well as elements of protection against forgery by certifying them with a digital signature of the issuer. But the creation and use of digital money in our legislation is not yet regulated, all operations with digital cash are carried out exclusively on agreements to use them as means of payment. This calls into question the protection of client data from fraudulent attacks, because if data loss occurs, it is difficult to prove that you have lost real money. We are not interested in the legal side of the issue, but in how the client can protect itself from data interception and, consequently, the loss of electronic money.

The largest system built on the basis of electronic wallets is Webmoney Transfer. For the calculations, the system uses accounting units, the so-called title units, which are analogous to money for the user (WMR, WME, WMZ, WMU, WMY, WM-C and WM-D). WebMoney allows you to make transfers only between wallets that have the same currency.

The second most common and popular payment system is Yandex.Money which is based on the technology of the payment system PayCash, which uses the model of digital cash. Instead of electronic coins, PayCash uses its development, the “payment book”, the face value of which is confirmed by the bank’s signature. To protect “payment book” the electronic coins are made disposable, that is, when making a payment, the bank checks whether this money has not been used before.

PayCash allows you to store digital cash directly in customers’ electronic wallets, completely independent of the server system. Such an approach provides enhanced protection of the system itself, however, if the electronic media on which the cash is recorded is lost or malfunctioned, the client’s electronic money is also lost. But in terms of security, it surpasses traditional payment systems, such as payment cards [3].

RUpay system allows you to make instant internal transfers between users, international and domestic Russian bank transfers, accept and send funds to e-wallets of other electronic payment systems, exchange electronic money of different systems, organize payment acceptance in a variety of ways on your Internet resource, receive a virtual payment card Visa. The client can manage this virtual card through the PayPal service, but its operation in Russia is limited, as it is used only for payments for purchases.

4. Security problems of payment systems

In the case of network money, a compromise of passwords and security codes is enough, as a result of an attack on a user's computer, and money can be stolen. In addition, computer malfunctions can lead to the loss of obtained security certificates giving access to the e-wallet. The security instructions in all payment systems suggest certain rules, the observance of which significantly reduces the risks of electronic cash losses. For example, Webmoney offers additional authentication via mobile phone or email, as well as the use of a one-time key that is generated each time you log in to your personal.
account. But data interception allows an attacker to perform many operations without additional confirmation.

The anonymity that the issuer provides, without tying its signature to a specific digital bill, becomes a vulnerability for a user who may lose money due to an attack on the device from which he is entering the payment system. Even more dangerous exits through social networks.

5. Attacks at the authentication stage

Authentication is a necessary procedure for accessing a payment system. The service used by the client assumes the generation of keys and one-time passwords. It is at this stage that an attacker can begin to obtain the necessary data to break into an electronic wallet [4].

You can get a user login without any problems - interception of the session identifier by the fraudster will give him information about the user.

When a user enters his e-wallet, a number is assigned to him (this number is saved in the browser's cookie and transmitted to the site when each page is opened) - in order to save information about actions in this session. The server will store the file that is responsible for identifying the user. If the fraudster is able to get the current session number, he will be able to impersonate the client and get a lot of data.

A cryptographic token will protect against the interception of this information when an attacker attempts to become an intermediary between the client and the electronic wallet. Approximately the attack looks like this:

Having settled on the communication channel between the User and the electronic wallet, the fraudster intercepts the User's requests and sends them to the site on his own behalf.

![Figure 1. The process of getting the session file.](image)

The fraudster broadcasts all the actions of the User and the site defines him as his client, as he enters the necessary password that only the User can know. In this case, it is impossible to determine that the client is not the one for whom it claims to be, since the source IP is identical all stages of the protocol.
To complete the attack, it is enough for an attacker to give the user some HTTP error and break the connection. The client will perceive this as a problem with the server.

In order to provide protection against such an attack, you must initially protect yourself from the threat of interception of the connection. Our solution is a cross-platform client application that will act as an intermediary between the wallet and the client.

```php
/* протокол авторизации НАС */
if (!isset($_REQUEST['session_name'])) { echo "Сессия не сформирована"; exit; }

session_start(); //старт сессии
require "config.php"; //подключаем конфигурационный файл

/* Проверка получена ли ответ в допустимом временном интервале*/
if ((time()) - $_SESSION['timestamp'] > $access_interval) { echo "Получен временной интервал ответа"; exit; }

/* Извлечение хея пароля из BD*/
if (!isset($_SESSION['password'])) { //подключаем файл с функцией для извлечения хея пароля
if (validate(Hash) == "NULL") { echo "Пользователь с таким именем не зарегистрирован"; exit; }

if (md5($md5($k$_SESSION['timestamp']).$_SESSION['ip']).Hash) == $x_type) { echo "Вы авторизованны"; exit; }

echo "Пароль не верный";
}
```

**Figure 2.** Form creation process.

**Figure 3.** Login extraction process.

6. **Cross platform**

Today, almost everyone in the world has mobile phones. Of the almost 7.5 billion people, more than 80% use phones and about 40% of them make purchases and pay bills through mobile phones and applications, and this makes it important to protect banking transactions and money transfers not only on personal computers, but also on mobile devices. To do this, it is proposed to implement cross-platform software that will reside on a USB drive and, when connected to a device, automatically determine its type and install the necessary software protection system, or, in case the security system is already installed, update the base of existing cryptographic protocols to the latest version. To connect to a mobile device, you need a Usb-Microusb adapter, which is available in stores. Next, we will
consider cryptographic protocols and security algorithms that we propose to use in the described software implementation.

7. PGP encryption
As a security measure for authentication, a pair of keys is used [5]. Exactly how the key is protected and determines its effectiveness.

For authentication, it has the prospect of PGP encryption. A PGP user creates a key pair: public and private key. When generating keys, their owner (name and email address), key type, key length and its expiration date are specified. The public key is used to encrypt and verify the digital signature. The private key is for decoding and creating a digital signature [6].

![Figure 4. PGP encryption algorithm.](image)

The application acts as a trust protocol, which confirms that the key sent by the User really belongs to him. That is, there is not enough information about the client ID, the correct login / password pair. The application in this case is the third party who confirms that no data has been changed.

For this, the application will generate one-time passwords for access to the private key [7]. That is, it will be impossible to create a digital signature of a user without having access to a one-time password. Each time you request a passphrase, you will also need to enter a one-time password - you need to add it directly to the end of the passphrase (without separating it with a space or any other characters). The entered password is immediately canceled: so, if the user makes a typo in the passphrase of the private key, then the next time you try to enter it, you will need to enter the next one-time password. When the password list is empty, any operations with private keys will become unavailable until a new list is generated.

8. Cryptographic token operation scheme
The listing looks like this (based on Python) in simple question / answer form:

```
<len_p> | <len_j> | JSON(<header>, <type>, <fields>, <files_meta>) | [binary]
```

- len_p (8 bytes): Total packet length.
- len_j (8 bytes): JSON-packet length.
- header (list): auth authentication token (optional) and version application version.
- type (str): package ID.
- fields (list): Arbitrary set of fields.
- files_meta (dict): display file_path->file_length.
- binary (bytes): Concatenated file contents (optional).

Remote input of passphrases is implemented on the basis of a specialized Pinentry2 spacer application and performed using the following protocol:
server> gpg-agent: sets in the environment variable PINENTRY_USER_DATA (transmitted through the entire call stack gpg> gpg-agent> pinentry) information for establishing the IPC channel, including the name of the IPC socket and the authentication session key.

gpg-agent> pinentry: calls pinentry, passing the environment variable PINENTRY_USER_DATA and initializes the Assuan protocol.

server> pinentry: sends an open network socket of the client connection via the IPC channel (UNIX socket) directly to the pinentry process.

pinentry> client: sends the necessary data (text strings and launch options received from the gpg-agent in step 2) to the standard pinentry application via the provided network connection.

client: calls the standard pinentry application and receives a custom response (as an Assuan protocol response).

client> pinentry: forwards the user response.

pinentry: executes "alarm" commands if the entered password phrase triggered any of them.

pinentry> gpg-agent: reproduces the user response in the Assuan protocol started in step 2.

If you enter an incorrect passphrase, loop 2-8 will be repeated again and again as the application requests.

9. One-time passwords

One-time passwords are an extra security measure to protect private keys. After its activation, when using the private key, the user will have to enter a short random password (from a previously generated list) along with the main password phrase. Since for each operation such a password is unique, it prevents the use of the private key by the enemy, even if he intercepts the passphrase [8].

To use this function, activate it in the server settings file or with the --otp option when starting the server. After that, call the server with the option --gen-otp and specify the number of one-time passwords you want to generate. The longer the list, the less often it will have to be updated, but the more operations the enemy can perform if this list is compromised. At any time, you can generate a new list of passwords, and all passwords that remained in the previous list will be canceled; after generating a new list, restarting the server is not required.

If the OTP function is enabled, then each time you request a passphrase, you will also need to enter a one-time password - it must be added directly to the end of the passphrase (without separating it with a space or any other characters). The entered password is immediately canceled: if the user makes a typo in the passphrase of the private key, then the next time you try to enter it, you will need to enter the next one-time password. When the password list is empty, any operations with private keys will become unavailable until a new list is generated.

10. "Anxious" commands

It is possible to configure an arbitrary number of so-called "alarm", or emergency, commands. These commands are executed on the server side if a certain passphrase is entered into the pinentry2 dialog.

The rules for the "anxious" commands are set by special items in the server configuration file. The name of the rule must be unique and begin with the prefix "panic_". The value of the rule must consist of a protective token and then through the space of a shell command or several commands in the usual form. The protection token is a PBKDF2-hash of a passphrase that should trigger the rule. To get such a token, you must start the server with the --gen-token option and enter the passphrase and, if necessary, the number of hashing iterations.

When executing shell commands, among other things, the following environment variables are passed to them:
1. GPG_REMOTE_PID: GPG Remote Server Process ID.
2. GPG_REMOTE_KEYRINGS: Space-separated list of paths to non-empty gpg keychains.

One passphrase can trigger any number of rules if all of them use this phrase. A triggered command is invoked by the pinentry2 server process without returning any output. The command is called before
transferring the entered password to gpg-agent with the rights of the user from whom the gpg-agent process is started and the rules are executed in the order they are specified in the settings file.

The key is considered to be closed in the gpg-agent memory before it calls pinentry2, so using the rm/wipe commands to erase keychains in an emergency will not immediately destroy the key — you also need to stop the GPG Remote server process to prevent it from sending the result gpg back to the client. In those cases where there is a need for this, it is worth using by assigning to them identical security tokens or passphrases.

Instead of shell-commands, the "anxious" rules can contain only one special command:
1. STOP: Normally stops the GPG Remote server. The server sends an undefined error message to the client, completes processing of the remaining parallel requests, deletes all temporary data received from the client, and stops the program.
2. KILL: Immediately interrupts the operation of the GPG Remote server. The server sends itself a SIGKILL system signal without performing any final procedures.

A separate whitelist.conf file is created for it in the same directory as the configuration file. Its format is quite simple and is created with the following nuances:
1. Lines that do not begin with a dash and a space are ignored.
2. Each line contains one set of options.
3. A set can be represented by one option in short (with one dash) or in long form (with two dashes), or several options separated by a space in short or long form (in random order).
4. If the set contains words that do not begin with a dash, they have the following meaning:
   (a) The word in square brackets is an arbitrary parameter — the options in this set are considered parameterizable.
   (b) A word without square brackets is an allowed parameter value — options in this set are considered parameterizable and can accept a parameter only in the specified value. If you need to allow multiple parameter values, you must enter them in the same line, separated by spaces (quotes and escaping spaces are supported)
   (c) The word #NO_FILES in square brackets sets the flag "no files" for this option set - the program will not consider the command line arguments passed as file names.

11. Allowed list
Although entering an additional one-time password increases the security level of the connection, however, authentication takes place outside this application. In order to exclude any external influence it is necessary to create an “allowed list” of options.

A separate whitelist.conf file is created for it in the same directory as the configuration file. Its format is quite simple and is created with the following nuances:
1. Lines that do not begin with a dash and a space are ignored.
2. Each line contains one set of options.
3. A set can be represented by one option in short (with one dash) or in long form (with two dashes), or several options separated by a space in short or long form (in random order).
4. If the set contains words that do not begin with a dash, they have the following meaning:
   (a) The word in square brackets is an arbitrary parameter — the options in this set are considered parameterizable.
   (b) A word without square brackets is an allowed parameter value — options in this set are considered parameterizable and can accept a parameter only in the specified value. If you need to allow multiple parameter values, you must enter them in the same line, separated by spaces (quotes and escaping spaces are supported)
   (c) The word #NO_FILES in square brackets sets the flag "no files" for this option set - the program will not consider the command line arguments passed as file names.

12. Conclusion
Thus, we see that the user of payment systems is virtually unprotected against intruders. But, since we know the scheme by which data leakage is possible, we can protect users with an application that provides two levels of security: entering a one-time password at the beginning of a session, which is stored on an external device, and is not stored in the cached data and the creating «whitelist», which will provide access to the session only verified addresses.

References
[1] Russian Federal Law of 27.06.2011 N 161-FZ
[2] Poruleva E, Gareeva G, Grigorieva D 2018 Symbol of Science 1-2
[3] Musalaeva S 2010 Electronic money and payment systems PSE 4
[4]  https://blog.cryptographyengineering.com/2012/06/21/bad-couple-of-years-for-cryptographic/
[5]  Ateniese G, Fu K, Green M, Hohenberger S 2006 ACM Transactions on Information and System Security (TISSEC) 9 1
[6]  Standage T 2007 The victorian internet : the remarkable story of the telegraph and the nineteenth century's on-line pioneers (New York: Walker) p 119 ISBN 9780802716040
[7]  Richter A, Wood J 2016 Chapter 3: Authentication Methods Practical Deployment of Cisco Identity Services Engine (ISE) pp 39-47
[8]  Regis J 2015 Bates Chapter 5: Authentication Securing VOIP pp 105-122
[9]  Ruiz-Martínez A, Cánovas Reverte Ó, Gómez-Skarmeta A 2012 Computer Standards & Interfaces 34 1 pp 80-92
[10] Ogbanufe O, Kim D 2018 Decision Support Systems 106 pp 1-14
[11] Dragusha M K, Kosumi A 2013 IFAC Proceedings 46 8 pp 44-49
[12] Tsiakis T, Stephanides G 2005 The concept of security and trust in electronic payments Computers & Security 24 1 pp 10-15
[13] Chang Ch-Ch, Chang Sh-Ch, Lee J-S 2009 Computers & Electrical Engineering 35 5 pp 757-763
[14] Halunen K, Häikiö J, Vallivaara V 2017 Pervasive and Mobile Computing 40 pp 220-241