QR-CODE BASED NON-REPUDIATION TRANSACTION VERIFICATION SYSTEM

Jakub Nantl[1]

[1] *Silesian University in Opava, School of Business Administration in Karvina, Univerzitní nám. 1934/3, 733 40 Karviná*
Email: jakub.nantl@forever.cz

Abstract: The state-of-the-art methods for securing e-commerce transactions resistant even to compromised client workstation are based on using independent security devices or separate communication channels. These systems either do not support non-repudiation and/or are expensive and thus suitable only for special cases, where such additional costs are justified. In this article a new cheaper transaction verification method, which supports non-repudiation, is described. A cell phone or tablet communicating with the client workstation using qr-codes is used as an independent security device. There are no additional costs for service providers except for initial SW implementation. The proposed method brings better transaction verification to the areas where the usage of current methods is uneconomical.

Keywords: bar-code, e-commerce, man in the browser, man in the middle, non-repudiation, qr-code, transaction verification.

JEL classification: L86

Doručeno redakci: 2.10.2012; Recenzováno: 3.3.2013; 15.2.2013; Schváleno k publikování: 11.9.2013

Introduction

After successful exploration of man-in-the-middle attacks (MitM), various techniques and counter measurements were deployed to protect the communication channel, making such attacks obsolete. Man-in-the-middle attacks just did not disappear but especially application and implementation flaws are exploited today, not the protocol itself. With communication channel secured, attackers are focusing on another weakest link of the chain: users' workstations.

Typical users' workstation software includes the operating system and several user applications usually from several vendors. A highly complex software compiled from millions of lines of source code is prone to errors. As there is no chance for ordinary user to inspect all his/her installed software, s/he must trust his/her software vendors that there are no known security problems and that if discovered, they will be promptly fixed. Unfortunately this does not work. As we can see, numerous security incidents are affecting not just individual IT users, but big corporations (Zetter, 2010) and government agencies (Langner 2011), (Farwell and Rohozinski, 2011) with their well-funded IT security departments.

Once we accept the idea that we are not able to secure our workstation, we should assume it is not secure as we expect communication channel to be insecure. New types of security mechanisms should be deployed and used to mitigate such problems.

1 Man-in-the-browser attack

After gaining control of end user workstation, attacker can install passive malware, such as keylogger or activity monitor that can monitor and log all user actions, successfully revealing user passwords and sensitive data. But he can also actively interfere with ongoing communication, successfully circumventing vast majority of known used security methods by intercepting data after they are decrypted and just before they are rendered to user display and vice versa.

The man-in-the-browser attack (MitB) is a modification of the man-in-the-middle attack, which does not target the communication channel outside communicating nodes secured by strong cryptography, but communication inside the application.

Encrypted and signed incoming data from the other communication endpoint is decrypted and verified by the encryption layer of application and then forwarded in cleartext to other application modules for processing and displaying. The same mechanism is used for outgoing data: user input is processed inside the application by processing modules and only after processing it is encrypted, signed and transmitted to the other party.

The attacker can redirect data flows inside the application to his evil code by various programing techniques, such as remapping API calls, using browser extensions etc. He is now able to modify data on-the-fly and totally disrupt communication. Let us consider online banking application.

A bank client wants to wire some money. He launches browser, logs in to his bank, enters beneficiary details and amount and submits his request. But the request is not sent to his bank directly but is passed to an attacker module. It is easy for attacker now to change transaction details, since the request is still not secured by any means. He replaces the amount and the target account number and returns the modified request to the next application module and lets the browser to encrypt, sign and send it to the bank.

The bank will accept such a request and sends back the confirmation page with evil transaction summary. However, this page has to be rendered by a compromised browser too. The attacker will modify this transaction confirmation page before it reaches the display module, replacing evil the transaction detail with a good one and returns it to the browser, which will render it for him. The victim checks the displayed confirmation page, logs out and has no idea that his money is sent to the wrong account.

As we can see, this type of attack, which targets the computer and application itself and allows the attacker to confuse communication parties can circumvent all security mechanisms that solely depend on interaction with the PC workstation alone. The only security schemes immune to MitB attacks are schemes that do not rely on computer as the only interface to communication.

As we have shown, the vast majority of security schemes, such as passwords, one-time passwords, PKI certificates, smartcard readers (class 1-3), security tokens etc. do not provide any protection against MitB attacks since there is no way to verify what exactly our workstation is sending to the other end of communication and what exactly we are authorizing.

2 The second communication channel

One way to prevent MitB attack is to use second independent communication channel with independent interface. In this case attacker will have to compromise both channels to make attack successful.

The typical contemporary second channel scenario is based on SMS verification codes. User sends transaction request to the other party. Verification code is generated there and sent back to the user's cell phone and it is retyped by user to the browser. The verification message have to contain not just generated one time verification code but also transaction details. User is then able to verify what exactly the other end thinks he is trying to do. Evil transaction details will appear in verification message and user will simply not confirm it.

As there is a great penetration of cell phones in the population, this system is easy to deploy but has some weakness: it does not support non-repudiation and it generates additional costs for second channel traffic. Also we cannot consider cell networks secure anymore (Perez and Picó, 2011). Additional encryption and message signature is thus necessary but unfortunately is not used widely.

3 External device

There are a lot of types of external devices used for authentication and authorization but only those equipped with keyboard and display, that generate confirmation codes based on transaction details are useful in prevention of MitB.

When user enters requested transaction details to his browser, the same data is entered to the external authorization device. Unique authorization code is then generated by device with algorithms using transaction details as input and it is displayed on display. This code has to be retyped to the browser manually and then sent to the server together with transaction request. If the appended verification code and transaction request match, transaction is considered authorized.

This is indeed secure system, but is unfortunately costly (logistics, device price) and not very user friendly, since user has to retype long transaction details to the device using usually tiny keyboard. Non-repudiation is not supported since symmetric cryptography is used.

4 QR Code

QR-code is a two dimensional, two color barcode developed by Denso-Wave in 1994. It offers increased data capacity, rapid detection and decoding and several levels of error correction. It was originally developed and used for warehouse logistic but quickly adopted for the other use.

QR-code is de facto current standard for transmission of short texts to cell phones. It's widely used especially in Japan for various means such as vCard, URL or phone number exchange and everywhere where we need to make it easy for user to enter some data into his phone or PDA.

Table 1: Maximum QR-code capacity

Numeric code only	7,089 characters
Alphanumeric	4,296 characters
Binary (8 bits)	2,953 bytes
Kanji/Kana	1,817 characters

Source: http://www.qrcode.com/en/qrfeature.html

5 A proposed scheme

As shown above, there are methods to deal with the man-in-the-browser attack problem now. Unfortunately these methods are suitable primarily for big deployments especially for their additional costs and availability.

As a vast majority of cell phones and PDAs are equipped with cameras and graphical displays. Cellphone can be simply used as a cheap, widely available external authorization device. Using embedded camera for data input reduces annoying double input of transaction details. Utilizing widely adopted device of everyday use increases user comfort – no need to carry another device.

5.1 Pre-shared key

Let's assume, both communicating parties use pre-shared secret *K*, that was exchanged using secret channel. User enters transaction details *T* and send it to the server. Server generates unique transaction id *n* and encodes it together with transaction data *T* into QR-code presentation and sent it to the user as an image on authorization page. Image is then scanned by cell phone, decoded and displayed to the user.

Table 2: pre-shared key scheme notation

Notation	Description
K	Shared secret
T	Human readable transaction details
n	Transaction ID
H()	Hash function
R()	Reduction function
Eqr()	QR code encode
Dqr()	QR code decode

Source: own

If the displayed transaction details match, phone generates one time verification code by computing and reducing hash of transaction details *T*, pre-shared secret *K* and transaction id *n*. Verification code is displayed on device screen and is typed into the browser and sent to the server. Server computes the same verification code from the saved transaction details and transaction id and compares it with received one.

Table 3: pre-shared key scheme

client		**server**
T	=>	
	<=	Eqr((T+n)
P=R(H(T+K+n))	=>	
		P ~ R(H(T+K+n))

Source: own

Such method was already introduced (Starnberger at al., 2009) but due to nature of symmetric cryptography non-repudiation is not supported without using a trusted third party.

Number of transaction steps can be further reduced by implementing QR code generation directly into the browser. Browser itself generates QR code presentation of transaction details from entered data without contacting the remote server. After scanning and verifying it in the cell phone, the same reduced hash is computed and used as authorization code in transaction form. Both transaction details and authorization code are sent at once to the server. The same verification is performed on the server side.

As a whole necessary data is sent to the server in one request and no negotiation is needed, this reduced scheme is suitable for off-line and/or batch processing.

The purpose of transaction id is to disable replay attacks. This unique code can be either generated at server side or at client side, but in this case server has to perform additional uniqueness checks.

5.3 Non-repudiation

To provide non-repudiation services, a digital signature should be used for transaction authorization instead of hash based on shared secret.

Table 4: asymmetric key scheme notation

Notation	Description
T	Human readable transaction details
n	Transaction id (nonce)
S()	Digital sign function
V()	Digital sign verification
M	Message

Source: own

Like in the other asymmetric cryptography schemes, communicating parties have to securely obtain public key of the other end or decide to use PKI for mutual authentication initially. Transaction details *T* are sent to the server. Server digitally signs *T* together with transaction id *n* and returns QRcode encoded message to the user. Image is scanned and decoded by the phone, digital signature is verified and user is displayed transaction details *T*. If accepted, user's digital signature of *T* and *n* is generated and have to be delivered to the server.

Even if we use cryptographic schemes that produce short outputs such as elliptic curves, the result will consists of several hundred bytes. It cannot be retyped to the computer by hand. It is displayed on the phone screen as Qr-code image, scanned and decoded using webcam and send to the remote server. Digital signature is verified on server and the transaction is accepted or declined.

Table 5: asymmetric key scheme

client		**server**
T	=>	
	<=	M=T+n+S(T+n)
V(M)		
M=T+n+S(T+n)	=>	
		V(M)

Source: own

If the browser is capable of generating qr-codes itself we can again reduce the complexity of the operations. Transaction details *T* are digitally signed in the phone using user's private key together with unique transaction id *n*. Digital signature is transferred to the PC using qr code and sent to the server together with transaction details *T*.

The transfer of digital signatures from phone back to PC using qrcodes may be problematic in some scenarios. In this case, we can utilize another ability of all smartphones and tablets. It's the ability to directly connect to the internet either by wifi or by the mobile data connection.

The beginning of the transaction processing is the same as in the previous method: Transaction data *T* is sent from the PC to the remote server. Digital signature of *T* and random unique nonce *n* is returned to the PC and transferred to the phone by scanning qrcode. User validates or reject transaction by generating digital signature *S* of *T+n*. The result – digital signature *S* is sent to the remote server directly from the phone using wifi or mobile data connection together with nonce *n*. Remote server identifies session by nonce *n*, validates saved transaction data *T* against received digital signature *S* and informs user about the result on both phone and PC.

5.4 Security considerations

The only secrets in proposed schemes are shared or private keys. The shared secret is stored in the cell phone of the user and on the remote server. Both devices must be kept secret. The attacker that is able to get access to the shared secret or user's private key is able to validate any transaction.
Generated verification codes in both symmetric and asymmetric variants do not contains any transaction details. Unlike SMS codes, they are generated by the transaction initiator. They are also valid only for the one transaction and cannot be reused. There is no need secure its transmission.

6 Further work

I am focusing in utilization of qr-code based pc cell phone communication in other asymmetric key cryptography applications like PGP or SSH. My goal is to keep private keys secret by not revealing it to the workstation environment. All operations with private key are done on a cell phone and only the results are passed to the potentially insecure computer.

Conclusion

As we have been shown, the only secure transaction verification schemes that are immune to man-in-the-browser attacks are schemes that do not depend solely on client workstation and use transaction details at the same time. Widely used schemes based on the second communication channel such as SMS verification codes or external authorization devices are secure but costly and thus not suitable for everyone.

A camera equipped cell phone, tablet or pda can be used as an external authorization device and QR-code communication makes it easy to enter data to (or from) the phone. The verification process is therefore user friendlier and usage of cell phones makes the proposed transaction verification method cheaper to deploy and maintain than the others.

On the other side proposed transaction verification scheme is not a "holly grail" of transaction security. Mobile devices are becoming targets for attackers itself nowadays. If both computer and cell phone are compromised by the same or cooperating attackers we are not able to maintain security of the e-commerce system.

References

[1] STARNBERGER, G., L. FROIHOFER and K. M. GOESCHKA, 2009. QR-TAN: Secure mobile transaction authentication. *International Conference on Availability, Reliability and Security. ARES '09*, 578-583

[2] ZETTER, K, 2010. Google hack attack was ultra sophisticated, new details show.[Online] *Wired Magazine.* [cit. 19th September 2012]. Accessible from: http://www.wired.com/threatlevel/2010/01/operation-aurora

[3] LANGNER, R, 2011. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Security & Privacy* 9 (3), 49-51.

[4] FARWELL, J. P. and R. ROHOZINSKI, 2011. Stuxnet and the Future of Cyber War, *Survival*, 53(1), 23-40.

[5] PEREZ, D. and J. PICO, 2011. A practical attack against GPRS/EDGE/UMTS/HSPA mobile data communications. [Online] *Black Hat DC.* [cit. 19th September 2012] Accessible from: https://media.blackhat.com/bh-dc-11/Perez-Pico/BlackHat_DC_2011_Perez-Pico_Mobile_Attacks-wp.pdf

[6] FIELDING, R., J. GETTYS, J. MOGUL, H. FRYSTYK, L. MASINTER, P. LEACH and T. BERNERS-LEE, 1999. RFC 2616: Hypertext Transfer Protocol–HTTP/1.1. *IETF*. [Online] [cit. 19th September 2012] Accessible from: http://www.ietf.org/rfc/rfc2616.txt

[7] RESCORLA, E, 2000. RFC 2818: HTTP Over TLS. *IETF*. [Online] [cit. 19 September 2012] Accessible from: http://www.ietf.org/rfc/rfc2818.txt

[8] DIERKS, T. and E. RESCORLA, 1999. RFC 2246 The Transport Layer Security (TLS) Protocol Version 1. *IETF*. [Online] [cit. 19th September 2012] Accessible from: http://www.ietf.org/rfc/rfc2246.txt