Security Analysis of Mobile Phones Used as OTP Generators

Håvard Raddum, Lars Hopland Nestås, and Kjell Jørgen Hole

Department of Informatics, University of Bergen
Havard.Raddum@ii.uib.no, lma029@student.uib.no, Kjell.Hole@ii.uib.no

Abstract. The Norwegian company Encap has developed protocols enabling individuals to use their mobile phones as one-time password (OTP) generators. An initial analysis of the protocols reveals minor security flaws. System-level testing of an online bank utilizing Encap's solution then shows that several attacks allow a malicious individual to turn his own mobile phone into an OTP generator for another individual's bank account. Some of the suggested countermeasures to thwart the attacks are already incorporated in an updated version of the online banking system.

1 Introduction

There are many services on the Internet needing strong user authentication. User authentication is often achieved utilizing a two-factor authentication technique based on something the user knows, i.e. a static password, and something the user has, i.e. a one-time password (OTP) generator or a list of OTPs.

The Norwegian company *Encap* has developed a system enabling an individual to use his mobile phone as an OTP generator when authenticating to a web-based service. The phone runs a Java MIDlet, which communicates with a server to generate OTPs. This paper describes the two main protocols utilized by Encap's system in 2009 and analyzes their security.

Perhaps because the protocols were analyzed earlier [1], the authors' initial analysis only revealed two minor flaws in the protocol designs. Early in 2009, an online bank deployed Encap's solution as part of their customer authentication. System-level testing revealed that malicious software, or *malware*, on a customer's PC can steal sensitive information. An attacker can then use this information to turn his own mobile phone into an OTP generator for the customer's account.

Further testing revealed that a modified version of the malware attack can be directed against *all* customers of the online bank, including those who do not wish to use their mobile phones as OTP generators. We also found that a similar attack can be instigated using social engineering and a malicious proxy. Since the discussed attacks allow an attacker to use his own mobile phone to generate OTPs for a customer's account, the attacker can access the account whenever he wants until it is closed by the bank. We present countermeasures to thwart the described attacks.

P. Samarati et al. (Eds.): WISTP 2010, LNCS 6033, pp. 324–331, 2010.

The rest of the paper is organized as follows. Section 2 presents the protocols, Section 3 analyses the protocol designs, Section 4 describes attacks on an online bank utilizing the protocols, and Section 5 concludes the paper. There also exists a longer version of this paper [2] with figures of the protocols and a more detailed analysis.

2 Protocols Enabling Phones to Generate OTPs

This section describes the two main protocols used by Encap's system during 2009. Initially, the user downloads the Encap client (Java MIDlet) to his mobile phone. Then, the Encap client executes a protocol to register with both Encap's server and a service provider utilizing Encap's system for user authentication. After the successful execution of the activation protocol the user can run the authentication protocol an unlimited number of times.

Before describing the protocols, we make a few general assumptions. A protocol is aborted if a protocol step fails, e.g. to verify data. All communication channels between the parties of the protocols are protected with SSL/TLS, except for the communications between the user and his PC and mobile phone, as well as an SMS starting the Encap client on the phone. An Encap white paper [1] discusses the secure deletion of secret information on a phone after it has displayed a new OTP. We assume that both the secure deletion and the Java platform on the phone work as intended.

2.1 The Activation Protocol

After the user has downloaded the client software onto his mobile phone, he must activate the phone as an OTP generator before it can be used for authentication to a web-based service. The activation protocol takes place between five parties: the *user*, the user's *mobile phone*, the user's *PC*, the *Encap Server* (ES), and the *service provider* (SP). The main steps of the protocol are summarized below.

1. The user authenticates himself to SP using credentials already known to SP. (The authentication procedure may involve a 'traditional' hardware-based OTP generator.)
2. When the user asks to activate his mobile phone as an OTP generator, SP redirects the user's browser to ES with a URL that contains an activation request and a *Secure Object*.[1]
3. ES verifies that the Secure Object comes from SP, and gets the user's phone number.
4. ES sends an *activation code* to the user's PC and an SMS message to the user's phone asking it to start the client software.

[1] The exact content of the Secure Object in the URL redirecting the user's PC to ES is not known to us, but we assume it contains information enabling strong authentication of the SP to ES, and we know it contains information to identify the user.

5. The mobile phone asks the user to enter the activation code, available on his PC, and transmits the code to ES.
6. ES verifies that the activation code is the same as the one sent to the PC, and sends a challenge to the mobile phone together with an encryption key K_0. (The role of K_0 is explained in Section 2.3.)
7. The user chooses a personal identification number (PIN) and enters it on the mobile phone, which generates a *security code* and a *response*. The response is the encryption of the challenge using the security code as key. The security code and response are sent to ES, and ES stores the security code.
8. ES verifies that the response and the security code correspond to the challenge, and if so, the user has activated the mobile phone as an OTP generator for use with SP.

The activation protocol's main goal is to ensure that only the legitimate user's mobile phone is activated as the OTP generator for the SP. The protocol contains several steps to achieve this goal. First, the user must authenticate to the SP using an already trusted authentication mechanism. Second, the Secure Object authenticates the SP to ES. Third, the activation code sent by ES to the user's PC is sent back to ES from the user's mobile phone.

These steps should ensure that the PC and the mobile phone are in the same location, or at least that there exists a communication link between the person using the PC and the holder of the phone. Since the person using the PC is authenticated and has transferred the activation code to the phone, we can assume that this person really wants to activate the mobile phone as an OTP generator.

2.2 The Authentication Protocol

The OTP-based authentication protocol takes place between five parties: the *user*, the user's *mobile phone*, the user's *PC*, the *ES*, and *SP*. The main steps of the protocol are described below.

1. The user enters the identity he shares with SP on its login page.
2. SP asks the user for an OTP, and sends a request to ES to generate an OTP for the user.
3. ES first sends an SMS to the user's mobile phone to start the client software. It then sends a challenge to the phone together with two encryption keys K_i and K_{i+1}, whose role will be explained in Section 2.3.
4. The user enters his PIN on the phone, and the phone computes the same security code generated at the time of activation. The phone then encrypts the challenge with the security code as key and sends the ciphertext as a response to ES.
5. ES verifies that the response from the mobile phone corresponds to the challenge, and sends an OTP to the phone.
6. The user enters the OTP on the SP's login page, and SP contacts ES to verify that the OTP is indeed the correct one for this user.

The authentication protocol's main goal is to ensure that only the legitimate user can obtain an OTP from ES. The goal is achieved mainly because the phone's response to ES' challenge is the encryption of the challenge using the key (security code) made during activation. The correct generation of this key requires the correct PIN and other unique information, which only the person who activated the mobile phone is supposed to have. This person was in turn authenticated at the time of activation, hence we can be confident that he is the legitimate user.

2.3 Generation of Security Code and Responses

The hash function SHA-1 and the encryption algorithm AES with a 16-byte key are used to generate the security code and the responses. Hashing is denoted by $H(\cdot)$ and encryption with key K is denoted by $E_K(\cdot)$.

The security code, SC, is computed by the following hash, truncated to 16 bytes,

$$SC = H(PIN||IMEI||CR||SPID)_{16}\,, \tag{1}$$

where $||$ denotes concatenation. PIN is a secret number with at least four digits entered by the user; $IMEI$ is a 14 digit code uniquely identifying the mobile phone where the Encap client is installed; CR, or *client reference*, is a 40-byte random string generated on the mobile phone during the activation protocol; and $SPID$ is a public value identifying the SP to whom the user wishes to authenticate.

The client reference (CR) needs to be stored on the mobile phone for later use. It is only stored in encrypted form. During the activation protocol it is encrypted using AES with the key K_0 which is sent by ES together with the challenge.

When the client reference is needed in the authentication protocol it is first decrypted using K_i, and when it goes back into storage it is encrypted using K_{i+1}. The keys K_i and K_{i+1} are sent from ES together with the challenge in the authentication protocol.

The generation of a 16-byte response, R, to a challenge, C, is defined by the expression

$$R = E_{SC}(C)\,,$$

where SC is the 16-byte security code defined by (1) and C is a 16-byte challenge received from ES.

3 Possible Improvements to the Protocol Designs

Our analysis of the protocol designs suggested two minor changes. The generation of the security code should be upgraded in accordance with "best practice," and a few steps in the protocol specifications could be simplified without losing any security benefits.

3.1 Improved Security Code Generation

The security code defined by (1) is generated in the activation protocol. The code is used as a shared secret key between the ES and the client on the mobile phone. Currently, the security code is generated by the client alone and then transferred to ES. When generating a shared secret key between two parties, it is preferable that no single party can control the value of the key. Instead, we suggest to use an established key exchange protocol, for instance Diffie-Hellman key exchange [3].

3.2 Encryption of Client Reference

The expression (1) for the security code contains a random 40-byte client reference (denoted CR). The purpose of the client reference seems to be to increase the entropy of the input to the hash function in (1). The client reference needs to be stored on the mobile phone for future use. It is specified that the client reference should only be stored in encrypted form, with keys to encrypt and decrypt supplied by ES.

At first there seems to be some added protection from this encryption: An attacker who gets hold of a user's mobile phone can determine the IMEI number of the phone and read the memory where the client reference is stored. The SPID is publicly known. If the client reference was stored in cleartext, then the attacker could record these values, and exhaustively try all different PINs to generate the set of possible security codes. Determining the correct security code would then be no harder than guessing the user's PIN. However, this approach is not available to the attacker since the client reference is encrypted before it is stored. Thus, the attacker needs the decryption key before being able to generate the (relatively small) set of possible security codes.

Unfortunately, the ES supplies the needed decryption key *before* any authentication takes place. If an attacker gets hold of a user's mobile phone and is able to read the encrypted client reference, all he needs to do is to follow the authentication protocol to get the decryption key from ES. Hence, the encryption of the client reference does not add to the security of the scheme.

According to Encap, another reason for introducing pairs of keys K_i, K_{i+1} to repeatedly decrypt and re-encrypt the client reference is to ensure that no two phones can obtain the same sequence of OTPs from the Encap server. If an attacker can copy the memory of a legitimate user's phone to his own phone, both can be used to generate the correct SC. However, as soon as one is used the other will become useless because it will not receive the correct decryption key in the authentication protocol. On the other hand, there is no need to apply AES to achieve this goal. Since the keys K_i are random bit strings, they could simply be XOR-ed with the client reference. The use of XOR instead of AES simplifies the activation and authentication protocols.

4 Attacks on the Phone Activation in an Online Bank

Early in 2009, an online bank deployed Encap's solution as part of their customer authentication procedure. One of the authors opened an account with the bank and used his own PC together with browser-based tools to study authentication related messages.

4.1 Malware-Based Replay Attack on Customers Activating Mobile Phones

The goal of the following *malware-based replay attack* is to collect a victim's username and password, and to generate the victim's OTPs on a mobile phone of the attacker's choice.

First, the malware captures the victims' username and password when he logs on to his online bank. Second, when the victim starts the implemented activation protocol, the malware captures the URL containing the Secure Object. The activation procedure is not secured against replay attacks occurring inside a time window of a few minutes, nor is it tied to one particular IP address or SSL session. Consequently, the malware need not disrupt the user's activation process, but can just wait until the user has completed the activation and then transmit the URL to the attacker's PC. The attacker enters the URL, containing the Secure Object, into a browser. Finally, the attacker submits his own phone number to download, install, and activate the Encap MIDlet on his mobile phone.

This attack was tested on an online bank account belonging to one of the authors (we did not try it on other customers' accounts). We found that the attacker's activation of a mobile phone as an OTP generator automatically overrides any previous activation made by the user. The attacker then has an OTP generator that enables him to log into the user's account. Moreover, the user is not able to log in anymore since his OTP generator is no longer accepted by the Encap system.

4.2 Malware Attack on All Customers

The malware-based replay attack can be modified to obtain a *malware-based impersonation attack* targeting *any* customer in an online bank utilizing Encap's solution—assuming that all customers have the option to activate their mobile phones as OTP generators. In this case, the malware just waits for a customer to log on to the online bank. The malware then sends a request to activate a mobile phone as an OTP generator without the customer realizing what is going on. When the URL with the Secure Object is returned, it is forwarded to the attacker's PC instead of redirecting the user's browser to ES. The attacker utilizes the URL to activate his own mobile phone as OTP generator for the user's account. This attack is deemed practical, especially since there already exist malware that steals information and manipulate client-server communication.

4.3 Phishing Attack on All Customers

An attack similar to the malware-based impersonation attack can be carried out without client-side malware. Because the SSL protocol cannot thwart *phishing attacks*, i.e. combinations of social engineering and MitM attacks, customers can be tricked into connecting to proxy servers under the control of attackers [4]. This can happen because customers are unable (or unwilling) to verify server-side public-key certificates used by SSL.

To initiate an attack, an attacker can generate phishing e-mails asking customers to log on to a MitM proxy masquerading as the customers' online bank. There is ample evidence showing that many individuals receiving phishing e-mails enter their login credentials at fake web sites.

Once a customer has connected to the MitM proxy, it forwards messages in both directions between the customer's PC and the bank's central infrastructure. The proxy can read all messages, change their contents, and create fake messages, in particular, the proxy records the username and password transmitted by the tricked customer. The proxy can then generate a request to activate a mobile phone as an OTP generator and use the returned URL to activate the attacker's phone as an OTP generator for the tricked customer's account. A MitM attack will normally give the attacker access to an account only once. This attack is different because it lets an attacker generate as many OTPs as he wants on his own phone. He can therefore access an account *whenever* he desires until the account is closed by the bank.

4.4 Countermeasures

We suggest three countermeasures to thwart the described attacks. To protect against the malware-based replay attack, the activation process needs to be secured against the replay of old requests. The ES must ensure that each Secure Object is only used once. Also, it should not be possible to just activate another mobile phone as OTP generator for an account, if there already exists a mobile phone activated for that account. A manual process should handle this situation.

To also protect against the malware-based impersonation attack and the similar phishing attack, there is a need for a tighter control over the transition from an old OTP generator to a new phone-based OTP generator. At the time of writing, an attacker who gets hold of a valid URL containing a Secure Object need not have an old OTP generator for the account under attack to turn his own mobile phone into an OTP generator for the account. A solution here is to let the user enter an OTP from the old OTP generator into the mobile phone, instead of the activation code provided by the activation protocol. The ES must then verify this OTP with the SP. This additional step ties the holder of an existing OTP generator to the mobile phone that is being activated.

It should be noted that it is difficult to completely defend against the impersonation attacks since an attacker can create a fake web page to ask a customer for the extra OTP required by the suggested countermeasure [5]. The same technique will not work for the replay attack, since the ES now rejects any earlier received activation request.

5 Summary

The Norwegian company Encap has developed a system allowing individuals to use their mobile phones as OTP generators. We suggested two minor changes to Encap's protocol designs, one to bring the activation protocol's key generation in line with "best practice," and one to simplify the designs without reducing the security.

A third party was responsible for integrating Encap's product into the evaluated online bank. The integration enables several practical attacks. The described client-side malware and phishing attacks on the customer authentication in the online bank are possible because the defense against replay of old activation requests is insufficient, and because the link between the previously used OTP generator and the new phone-based OTP generator is too weak. Encap received an early version of this paper with recommendations to implement the suggested countermeasures to thwart possible future attacks. The authors have since been informed that Encap and the third party have implemented some of the described countermeasures. The details of the implemented countermeasures are not known to us.

The seriousness of the attacks shows how important a system-level analysis and testing can be to determine the level of security provided by protocols in a real system. Since the Encap solution is new, it should be further scrutinized for weaknesses. The authors believe it is particularly important to study how the Encap solution should be integrated into existing web-based services. It may also be interesting to study how to use the mobile phone to detect modification or spoofing of transaction requests from a customer's PC, see [2] for suggestions.

References

1. Hagalisletto, A.M., Riiber, A.: Using the Mobile Phone in Two-Factor Authentication, Encap white paper,
www.encap.no/admin/userfiles/file/iwssi2007-05.pdf
2. Raddum, H., Nestås, L.H., Hole, K.J.: Security Analysis of Mobile Phones Used as OTP Generators, Reports in Informatics, 392, The University of Bergen (2010), www.ii.uib.no/publikasjoner/texrap/pdf/2010-392.pdf
3. RFC 2631, Diffie–Hellman Key Agreement Method (June 1999),
tools.ietf.org/html/rfc2631
4. Jøsang, A., AlFayyadh, B., Grandison, T., AlZomai, M., McNamara, J.: Security Usability Principles for Vulnerability Analysis and Risk Assessment. Presented at the Twenty-Third Annual Computer Security Applications Conference (ACSAC), Miami Beach, FL, USA, December 10-14 (2007), www.acsac.org/2007/papers/45.pdf
5. Hole, K.J., Klingsheim, A.N., Netland, L.-H., Espelid, Y., Tjøstheim, T., Moen, V.: Risk Assessment of a National Security Infrastructure. IEEE Security & Privacy (January/February 2009), www.nowires.org/Papers-PDF/RiskEvaluation.pdf