Implementing Strong Authentication with OTP: Integrated System

Pance Ribarski, Ljupcho Antovski

University Ss. Cyril and Methodius,
Arhimedova bb, 1000 Skopje, Macedonia
{pance,anto}@ii.edu.mk

Abstract. Due to the arising problems with using static passwords, there is a strong need of implementing more secure protocols for authentication. The One-Time Password protocol is implementation for two-factor authentication; the two factors being something you own (a token) and something you know (PIN). This paper presents an open algorithm for OTP and implements a client-server system for secure OTP authentication. The implemented system is full-blown authentication system for multi-site secure authentication ready to be implemented in a real-case scenario.

Keywords : security, password, authentication, multi-factor, two-factor, one-time password, internet

1 Introduction

Problems with authentication online appear more frequently. The security level of nowadays static passwords does not overcome the threats rising in stealing those passwords [15]. There are numerous examples of stolen passwords, phishing scams and many other techniques that allow bad guys to take our passwords [14]. This disrupts the authentication process – process of confirming the identity. This paper introduces a platform for secure authentication employing two-factor algorithm. In chapter 2 we identify the problem. In Chapter 3 we discuss the changes organizations need to embrace in order to bring security to a satisfactory level. These changes have to be enforced in almost every place where classic password authentication is used. In Chapter 4 we review the One-Time Password paradigm and the general ways of implementation. In Chapter 5 we explain the developed working OTP algorithm – algorithm that is under the auspices by the open community of OATH [9]. Chapter 6 underlines our contribution to this OTP area – providing a whole system for secure authentication. Our solution consists of server side – that the authentication organizations should implement, and client side – that consists of software tokens installable on mobile phones.

© ICT ACT – http://ictinnovations.org/2010, 2010

2 Problem Identification

In computer security, authentication is usually an automated process of verifying the identity of someone or something, such as a computer or application. [19, 1]. Simply, the authentication process is the basis of every multiparty communication in which the parties need to introduce themselves. The introducing part is very difficult to conduct in electronic manner – the other side can never be sure if the introducing identity is the true one.

The classic approach is using a pre-shared secret, that acts as a challenge when the introducing party needs to prove its identity. If the introducing party answers the correct secret, then the identity is proven. This method is copied from times far before the computer age, when a person would have to recite a word or a phrase in order to confirm the right identity [20]. In the computer world this is called a static password.

Static passwords have are flaw in secure manner by design – they don't change over time. Even if they change, the new password must be distributed to the user for further use. Decreasing the life-time of a password has led to users writing their passwords down because they simply can't remember such short-timed passwords. The static nature of the passwords also leads to man-in-the-middle attacks where someone intercepts the password and then uses it to authenticate as the person which stole the password from. The man-in-the-middle scenario and some other password stealing scenarios can be performed variously [21]:

- network administrators or sniffers can sniff for password roaming around [22];
- trojan horses and keyboard listeners can steal passwords as they are typed [23];
- phishing attacks can lead the user to giving away their password [24];
- using social methods to extort a password from a person [25];
- brute-force lookup for a passwords – modern technology spread this way even on non trivial passwords [26].

3 Multi-Factors Authentication

The answer to these problems obviously is the authentication method where the user does not disclose static information. We want to authenticate answering a challenge that always changes. This way, even if someone is looking over our authentication process, it cannot use that information to authenticate as us later on. Even more, we must consider an authentication that requires multi-factors. These required factors ensure that if someone is going to authenticate as us, he must collect all of the factors. The factors that we can choose from are loosely categorized as [16]:

- biometric data – something you are; this factor can be fingerprint, retina scan, voice, or some other bio-information
- static password – something you know; discussed in Chapter 2

© ICT ACT – http://ictinnovations.org/2010, 2010

- hardware or software – something you have; this is supposed to be unique piece of data that cannot be copied

This research is trying to cover the level of security in the area where we already authenticate with static passwords. It is common sense that if we add more factors we get better security. This is wrong with the authentication process we try to employ – we try to better the plain password authentication and minimize the complexity of the process. This area will be completely satisfied with a two-factor authentication – the method we are proposing. From the list of categories from which we can choose a factor, we choose a static password and a hardware or software. This concludes to a two-factor system in which if we want to authenticate we need to know some password (further noted as PIN) and a device that will generate a random code every next authentication. This solves the problem with static passwords, and generally increases the level of security with increasing the number of factors to two. This kind of authentication is named as One-Time Password, or shortly OTP [27].

4 OTP implementations

There are various implementations of OTP systems [17]. All of them are with the same cause – generate a token that we pass to the authentication server. The authentication server generates one on their own – and checks if the two tokens are equal. If they are equal, than the authentication process is successful. The creation of these tokens is done in various ways:

- create a token using one-way function [18] taking a parameter from the previous token creation; this initial token is something that both the client and server know and is random by nature; randomness is provided from Java Crypto Class KeyGenerator
- create a token using one-way function taking a parameter from an accurate clock; the clock in the client and server must be synchronized
- create a token using one-way function taking a parameter from a secret key and a counter; the secret key is pre-shared with the client and server in an secured manner

There are implementations of OTP that are open – allowing users to see the algorithms and allowing the community to verify that it is secure, or if it is not secure to improve it [8, 10, 11, 12]. But most of them are commercial, meaning that the implementation is hidden and the usage of the system requires registration fees [2, 3, 4, 5, 6, 7].

The implemented systems are meant to be used as a way of two-factor authentication in various systems. They are implemented in different platforms, like C, Java, .NET. The user can generate the token in different ways: using smart cards [4, 6], USB tokens [5], or use the mobile phone as token generator [2, 3, 7]. Some systems are implemented to be used in the UNIX environment [10, 11, 12], thus securing the classical username-password authentication system in UNIX.

© ICT ACT – http://ictinnovations.org/2010, 2010

The Initiative for Open Authentication [9] is an organization founded by the industry companies trying to unify the authentication process. Their aim is to "provide reference architecture for universal strong authentication across all users and all devices over all networks" [29]. Their joint effort has produced many documents and protocols regarding strong authentication. Among them is the HOTP [8] algorithm which is counter-based one-time password scheme. We are using this scheme in the implementation of our OTP authentication system.

5 Implementation Algorithm

The algorithm proposed from the Initiative for Open Authentication generates numeric OTP values with length from 1 to 8 [8]. The greater the numbers, the stronger the authentication process. This algorithm requires a *symmetric key* (denoted *K*), known only to the authentication server and the client. The transportation of the key to the client should be done over a secure channel. The key transportation is done only one time – at the client token creation. This will be repeated only if the client token is lost, or malfunctioning. In this case it's better to create a new key, as if it is a new client token. Another key part of this algorithm is the parameter that changes over time – exactly every time a new password is generated. This parameter is called a *counter* (denoted *C*). At first the counter has a value of zero at both the client and server. When the client token generates password, the counter is increased by one. If the server is given a generated password for authentication, it generates a password on its own with the current counter for a given client, a check for the equality. There is a possibility that a client will generate a password and not use it. This way, the client counter will increase, but the server counter will not. If the client sends a password to a server, it will not match with the server's. In this scenario, the server has to compensate with the counter, increase the counter couple of times trying to reach the client's counter. The number of times the server increases the counter trying to compensate is denoted as *S – resynchronization parameter*. This value should be kept at minimum to avoid brute force attacks. Another parameter that is handy in avoiding brute force attacks or denial-of-service attacks is the number of wrong *authentication attempts - V*. This parameter will increase every time the server fails to authenticate a client password using the resynchronization parameter S. Like S, the value of V should be kept at the possible minimum to avoid guessing the password, and yet to give the user a chance to wrongly use the system.

The password generation is divided in 3 steps. The first step is to generate Hash-based Message Authentication Code – HMAC [13]. This operation creates a message digest of some text using a secret key (the mentioned key K). Any iterative cryptographic hash function can be used. Our system is currently using SHA1 as cryptographic function, but other functions are tested as a work in progress. The text that is HMAC'ed is the counter C. The result from this step is 20-byte string.

The second step is extracting a 4-byte string from the HMAC value. This is optionally done in two ways – using static offset, or using dynamic offset. The static offset is always extracting 4 consecutive bytes from the 20 bytes HMAC value. Thus the static offset parameter can be 0-16. The dynamic offset is calculated from the last

© ICT ACT – http://ictinnovations.org/2010, 2010

4 bits from the HMAC value. The 4-byte extracted value the base for generating a password.

The third step is converting the 4-byte string into a decimal number. This decimal number is then shortened to a desired length using modulus with 10^D where *D* is the *password length*. The result from this step is the generated OTP password that we send to the server for authentication.

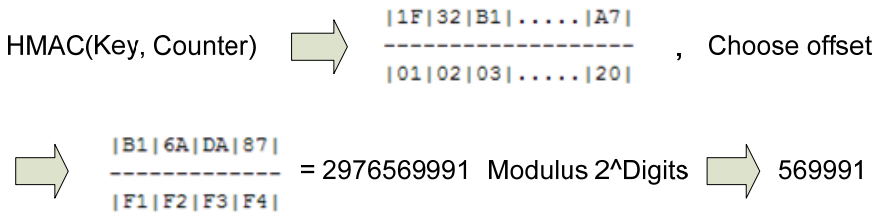

Fig. 1. Algorithm for creating OTP token

The security of this algorithm has been examined thoroughly [8]. The analysis shows that the best possible attack against the algorithm is a brute force attack. Even if attacker has access to consequential successful password values, building of a function based on these values does not have any more advantage over random guessing the values. This concludes that the brute force attack is the only option. Using brute force, the probability of guessing a password is:

$$p = \frac{S * V}{10^D}$$

In a system where resynchronization parameter S is 4, the number of wrong authentication attempts V is 5, and the length of OTP password is 7, the probability of guessing a password is only 0,0000035. This probability is in the acceptable boundaries for a secure system because a brute force attack with this probability rate is technically impossible.

6 The System

The OTP system itself that we are developing is the main contribution in the OTP area. This system is a unified platform consisting of server plugins and client plugins. Server plugins are the tools used at the authentication server. They receive requests for authentication and respond accordingly. The client plugins are tools to generate OTP passwords at client side, they are meant to be user-friendly, not expensive and generally usable. Systems like this already exist, but mainly as commercial products [2, 3, 4, 5, 6, 7]. These commercial solutions offer OTP authentication, but are closed in algorithm manner and users don't have insight in the inner working of the system.

© ICT ACT – http://ictinnovations.org/2010, 2010

This system allows registration of sites that will use OTP authentication. These sites have the abilities to manage users on their own. The main idea is that this system will work on its own, while all the administrative work will be done from the registered sites that are going to use it.

The server side consists of:

- database to keep all the information
- web site to manage all the administrative tasks
- web service for administrative tasks done by the registered sites and for delegating keys to clients

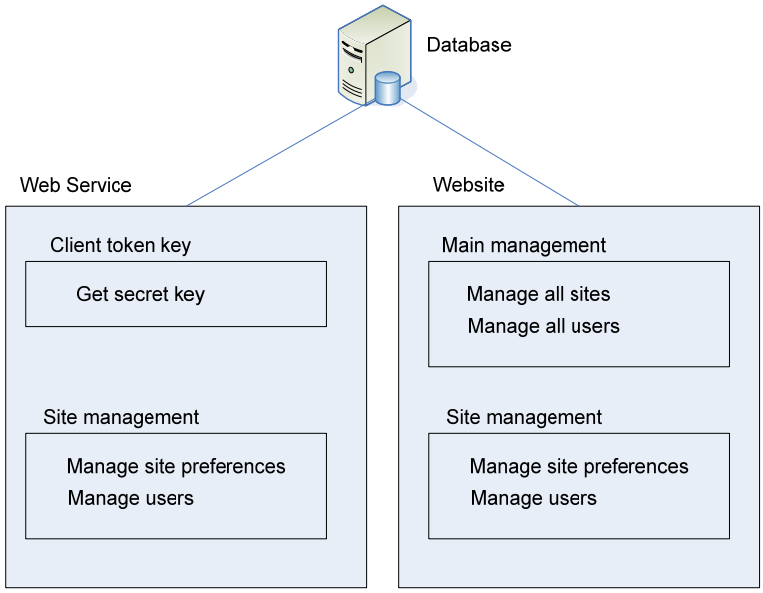

Fig. 2. Architecture overview of OTP system

The database keeps information about the registered sites and about the registered users. The main information for each user are its secret key and counter. These data are needed for the server to generate OTP password for requested authentication from a user. The web site is used for web management and administrative work with the whole system. It is mainly divided into system administrative part and administrative part for particular site. The system administrative part has permission to manage all the sites and all the users. Further, it can create new sites, and add new users to those sites. The administrative part for particular site can manage users only for their site. They can add new users, delete users, recreate a key for a lost device and more. The web service is part of the server tools. The main role is to distribute the secret key to the client token. This can happen only once, the first time a token in initialized. Further reading of the client key is forbidden. Exclusion to this rule is the case when a client token is lost and it needs to be recreated. But even then, a new key is created and this new key is only once distributed. The web service has also an administrative

© ICT ACT – http://ictinnovations.org/2010, 2010

role. Through the web service sites can manage their user databases. This is an acceptable way of interconnecting legacy systems at the sites' side with the OTP authentication system – programmers can use these web services and connect their own software infrastructure with the authentication system.

The client side is meant to be easy for the client, not expensive but still secure. The token hardware is the "something you have" factor from the two-factor authentication protocol. The only logical conclusion is to use the client's mobile device to act as OTP token generator [30]. Today's mobile devices are capable of running applications with limited resources. There are many platforms on which the mobile phones work: windows mobile classic, windows mobile 7, apple iOS, blackberry, symbian, android, java mobile and others. There is a great probability that any user has mobile phone built on some of these platforms. The system explained here is set to create software tokens – applications for every major platform. In that way the OTP will become easy and very cheap to implement in every organization where secure authentication is required. The token application is designed to operate with more sites – meaning that it holds keys from all the sites and the client is able to generate OTP passwords for many places with only one application. This contributes with the OATH community in a way that these mobile clients can be used with any authentication servers because of the open algorithms used. The clients can generate passwords for sites administered by the system explained here, and can generate passwords for any place that complies with the algorithm explained before.

The process of authentication is the following:

1. client enters authentication site
2. server sends OTP password
3. client checks if the password is correct; sends OTP password
4. server check if the password is correct; authenticates user

This double checking is a double OTP protocol. It is not usual with the OTP authentication protocols but strengthens the security in the process. This way we can avoid the biggest flaw in web authentication – phishing [24]. If we are actually authorizing at a site that pretends to be the site we want to authenticate on, this fake site will not generate a correct OTP password – it will not be equal with the password generated with our token generated password.

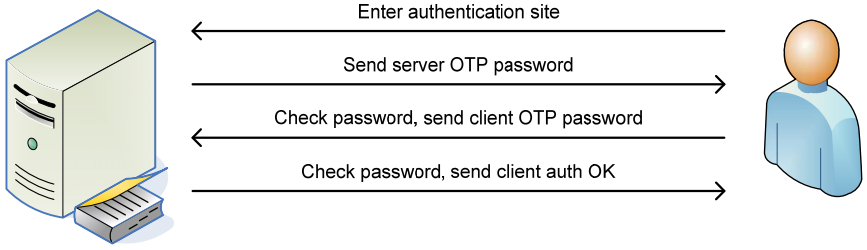

Fig. 3. Authentication protocol between client and server

© ICT ACT – http://ictinnovations.org/2010, 2010

7 Conclusion

The presented research is a work in progress towards an affordable an easy-to-use system for secure and strong authentication. The current classic authentication method – static password is prone to password stealing. The multi-factor authentication protocols are stepping in, One-Time Passwords are leading the way as currently popular protocol for strong authentication. The presented algorithm is easy to implement and its openness is giving the opportunity to have many developers embracing it in authentication systems. Our system is a solution both for the server side and client side. The server side is consisted of website and web service to serve to administrators and site managers. The client side is consisted of applications that can be installed on mobile devices which will act as hardware token generators. The whole system satisfies and prevents stolen-password, man-in-the-middle, denial-of-service and general phishing attacks. This system is a proof of concept that the transition to a secure authentication can be easily done, and that it can be user-friendly. The technological advantages of mobile devices give the opportunity of slicing costs for extra hardware tokens.

8 References

1. Ribarski, P., Antovski, Lj.: Introducing Strong Authentication for E-Government Services in Macedonia, 2009
2. FiveBarGate, http://www.fivebargate.net
3. FireID, http://www.fireid.com
4. Solid Pass, http://www.solidpass.com
5. Yubico, http://www.yubico.com
6. NagraID Security, http://www.nidsecurity.com
7. OTPXS, http://otpxs.com
8. HOTP, http://www.ietf.org/rfc/rfc4226.txt
9. Initiative for open authentication, http://www.openauthentication.org
10. Haller, N. M., The S/KEY One-Time Password System
11. OPIE – One-Time Password In Everything, http://wiki.linuxquestions.org/wiki/Opie
12. OTPW, http://www.cl.cam.ac.uk/~mgk25/otpw.html
13. Hash-based Message Authentication Code, http://tools.ietf.org/html/rfc2104
14. Dhamija, R., Tygar, J.D., Hearst, M.: Why Phishing Works
15. Fragkos, G., Xynos, K., Blyth, A.: The use of computers idle-time and parallel processing over a network to perform password threat assessment
16. Bhargav-Spantzel, A., Squicciarini, A., Bertino, E.: Privacy Preserving Multi-Factor Authentication with Biometrics
17. Takasuke, T., Akihiro, S.: One-Time Password Authentication Methods
18. Volpano, D.: Secure Introduction of One-Way Functions
19. VeriSign Identity and Authentication Glossary, http://www.verisign.com/authentication/information-center/authentication-glossary/index.html
20. The Roman Military System, http://ancienthistory.about.com/library/bl/bl_text_polybius6.htm
21. Sans Institute, SSL Man-In-The-Middle Attacks
22. Del Armstrog, Password Sniffing

© ICT ACT – http://ictinnovations.org/2010, 2010

23.Sans Institute, First Step Data Capture – Key Stroke Loggers
24.WordSpy Phishing, http://www.wordspy.com/words/phishing.asp
25. What is social engineering, http://www.microsoft.com/protect/terms/socialengineering.aspx
26.WASC Threat Classification – Brute Force, http://projects.webappsec.org/Brute-Force
27.One-Time Passwords Specifications, http://www.rsa.com/rsalabs/node.asp?id=2816
28. Java Crypto KeyGenerator, http://download.oracle.com/docs/cd/E17476_01/javase/1.4.2/docs/api/javax/crypto/KeyGenerator.html
29.OATH Features and Benefits, http://www.openauthentication.org/features
30.BearingPoint, Multifactor Authentication: A Brief Selection Guide

© ICT ACT – http://ictinnovations.org/2010, 2010