A Novel Web Security Evaluation Model for a One-Time-Password System

Ben Soh and Aaron Joy
Applied Computing Research Institute
Department Of Computer Science and Computer Engineering
La Trobe University, Bundoora Victoria 3083 Melbourne
{ben, awjoy}@cs.latrobe.edu.au

Abstract

One-time passwords (OTPs) have the advantage over regular passwords in that they protect legitimate users from replay attacks by generating a different password for each time of authentication. There are two variables that play a major role in creating a secure OTP; they are the passphrase length and the number of times the one-time password should be hashed. It is already a known fact that the larger the passphrase length the better is the security an OTP can offer. However, there is still a lack of quantitative analysis carried out to study how optimal web security can be achieved. To this end, we propose a novel web security evaluation model that can be used to measure the strength of a one-time password.

1 Introduction

Virtually all multi-user, ubiquitous systems require that a user provides a username and also a password [1]. This is because the frontline defense against intruders is usually an access control mechanism using password authentication.

There are many client/server applications that use passwords for authentication, ranging from basic telnet to remotely logging onto a supplier's web database. In all these applications, it is possible for an attacker to intercept the password and then replay it to the server. This replay problem can be overcome by using a system called one-time-password system (OTP) [2].

An OTP system has an advantage over a regular password system in that the former generates a different password for each time of authentication. In the one-time-password system, the password entered by the user does not traverse the network. This enables OTP systems to protect against passive attacks [3].

We have already designed and implemented an OTP system in our secure client/server, e-commerce framework [4]. A brief description of the OTP system is given in Section 2. The main focus of the paper is, however, on the security evaluation study of the OTP system, which is still lacking to the best of our knowledge.

2 Brief Description of the OTP System

The one-time-password system has two functions for the client/server, e-commerce framework: i) it provides authentication at the user level and ii) it provides a link with the next stage of authentication at the network level involving the use of public keys.

OTPs have two main variables: the passphrase length and the number of times the one-time password should be hashed. It is already a known fact that the larger the passphrase length the better the security it can offer [2,4,5]. However, the longer a passphrase, the more unwieldy will be for the user to remember. Therefore, a consensus is required for optimal web security and usability of the OTP system.

2.1 OTP Authentication

The OTP system carry out four stages of client/server communications for authentication purposes:

- **Stage 1 - clientRequest:** When a client request with a username is accepted by the OTP system, a variable called *challengeOTP* is created. The *challengeOTP* contains the random seed and the value N-1, where N is a sequence number set by the OTP system and is also the number of times a passphrase is hashed. Together with the user passphrase stored on the OTP system, the random seed is hashed N times to output a one-time password value, which is stored on the server.

- **Stage 2 - challengeOTP:** The *challegeOTP* is transmitted to the client. The client then asks the user to enter their passphrase. Subsequently, the public key, the random seed, and the passphrase are hashed N-1 times to produce a one-time password on the client machine.

- **Stage 3 - responseOTP:** This stage involves the client sending the OTP system a variable called *responseOTP*, which contains the username and the one-time password created in the previous stage. The OTP system then hashes the (N-1)-hashed one-time password from Stage 2 - once.

- **Stage 4 – verified:** The OTP values produced in Stage 1 and Stage 3 are compared. If the two values match, then the client is deemed to have been authenticated by the OTP system.

3 Performance Issues Involved in the OTP System

An OTP system is not flawless. For instance, if during the *OTPchallenge* stage, an attacker intercepts the hashed seed and then blocks access to the server's port, the server will crash. When the server restarts, the attacker could either take the client offline or race them to authenticate again to the server .

Another issue with the OTP system is a plaintext attack, which is descried below:

(a) An attacker employs a packet-capturing device, e.g. a software sniffer or a specialised piece of hardware. The basic set-up would involve the packet-capturing device being placed between the client and the server.
(b) The attacker waits for a *clientRequest* from the client and then captures the username of the client.
(c) The attacker waits for the *challengeOTP* from the server and then captures the *challengeOTP* and has obtained the N value and random seed as a result.
(d) Next, the attacker waits for the *responseOTP* to be sent and. Once the *responseOTP* is captured, the attacker can then carry out a brute force cracking of the OTP.

In the above attacks, the larger the value of N, the longer is the lifetime of the password. Therefore, a small N value implies that the OTP has to be reseeded more often. Frequent reseeding of the OTP will lead to administrative overheads. On the other hand, an OTP with a shorter lifetime will have less time for an intruder to crack. Each time a hash of an OTP takes place, CPU time and resources are consumed on the clients machine and on the server. This means the higher the N value, the larger is the consumption of CPU time and resources. Thus, a number of questions related to the OTP system performance arise: (i) What is good passphrase length? (ii) How can a good N value be determined? (iii) What is a good lifetime for an OTP?

4 Web Security Evaluation of the One-Time-Password System

The security testing was based on the speed at which an OTP can be broken. The experimentations were run on a Pentium (200MMX with 64mb of EDO ram) using Redhat Linux 7.0. The Unix time command was used to time the speed of the password cracking.

4.1 Experimental Method

The OTP system experimentation program consists of two main modules called *OTPCreation* (from our actual OTP system) and *OTPCracker*, which were both written in g++. The *OTPCreation* module executes the tasks of generating one-time passwords and client/server OTP authentication. On the other hand. the *OTPCracker* module is used to crack an OTP by brute force. The *OTPCracker* has two major sections: the actual cracker and a word generator. (Note: Due to space constraints, the source code for the *OTPCreation* and *OTPCracker* modules are not included in this paper.)

The word generator creates the passwords for the OTP cracker. In our experimentations, the OTP keyspace comprises 62 alphanumeric characters. The word generator works as follows: (i) When the count is incremented by 1, the character position that corresponds to the count in a character array is then displayed. (ii) If the count has reached the end of the array then an overflow is set to 1. When more than one character is needed, the overflow is just fed into the increment of the next module.

The top level of the experimentation program is summarized below:

1. Create a random user passphrase with a given length and an N value of 1 using the *OTPCreation* module.
2. Using the Unix time command to time the execution of the *OTPCracker* module.
3. Record the Unix user-time when the OTP is cracked. The user-time is the actual time the CPU has spent on executing the *OTPCracker* module.
4. Repeat step 1 with the N value incremented by a factor of 10 each time until N=10,000.

5 The Proposed Web Security Evaluation Model

To find out how secure a set of parameters for an OTP is, we will propose to evaluate how long it would take to break an OTP, given the value of N, computer speed, keyspace, and passphrase length. This would provide an OTP system with a means to measure the strength of a passphrase chosen by a user. To this end, we incorporate into the evaluation model two mathematical equations to predict the strength of a passphrase on the OTP system, within a certain level of confidence.

The first equation involves the time (*TimeToCrack* in seconds) required to break an OTP. A standard brute-force passphrase attack is of the form*: TimeToCrack = Combinations / TriesPerSec*, where *Combinations* is the number of possible passphrase words of a given length generated from a given keyspace, and *TriesPerSec* is the number of word comparisons a machine can perform in a second. However, since the first character of a passphrase is first compared in an OTP system, a more accurate value of *TimeToCrack* can be calculated as follows:

$$TimeToCrack = \frac{(1stcharpos) \times (keyspace^{(length-1)})}{TriesPerSec}$$

where *1stcharpos* is the numerical position of the first character of the passphrase with respect to the number of characters allowable for the passphrase, i.e. the *keyspace*, and *length* is the length of the passphrase

The second mathematical equation involves the N value, and *TriesPerSec*. To obtain this second equation, we need to use the results from Section 5 to find a relation between the N value and TriesPerSec. This is done by calculating the average number of tries per second over all possible passphrase lengths within the computational limitation of the machine. Table 1 shows the average number of tries per second for N = 1, 10, 100, 1000, and 10000, and their $\log_{10}$ values, and the results are presented in Graph 1.

From the trendline in Graph 2, a fourth order polynomial of *x*, where *x = N x 10*, is found to be the best fit for the curve produced in Graph 1. Based on this, the second equation can be derived as follows (for $1 \le N \le 10000$):

$$\begin{aligned}\log_{10}(TriesPerSec) = {} & (0.0075(\log 10(N \times 10))^4 \\ & - 0.0599(\log 10(N \times 10))^3 - 0.0432(\log 10(N \times 10))^2 \\ & + 0.2943(\log 10(N \times 10) + 3.5822)\end{aligned}$$

5.1 An Analytical Example

Let us give a simple example below to illustrate the above. Without loss of generality, we want to find out how long it will take to break an OTP if the N value is set to say 100 and the passphrase is set as “9okay”. In this case, the passphrase length is 5, the first character “9” is the 61st character in our keyspace containing 62 characters (see Section 4.1). Therefore, *1stcharpos* = 61 and *keyspace* = 62 and *length* = 5. Next, from the second equation with N =100 we are able to calculate *TriesPerSec*, which is 1165.467. Finally, from the first equation it can easily be calculated that *TimeToCrack* = 773386.5446 seconds. Using the same experimental OTP system, *TimeToCrack* for the same passphrase is found to be 773386.5109 seconds with N = 100. However, the equations should be seen as a very close estimate of predicting how long it would take to crack a passphrase in an OTP system.

6 Conclusion and Future Work

From the analysis of the actual OTP system designed for our E-commerce system framework [4], we have found that there are two major factors influencing the time to break the user passphrase. They are the N value (the number of times the OTP is hased) and the passphrase length. Based on the experimental results obtained, we have been able to fit a trendline to derive two analytical equations (see Section 5). These mathematical equations provide the OTP system operator with the ability to predict the strength of a passphrase chosen by a user. The OTP system would then be able to assign an N value to optimize the security of the OTP generated from the user passphrase.

The future work includes extending our prototype OTP system to a full-scale OTP server using different types of machines on a variety of platforms.

8 References

[1] William Stallings, *Network Security Essentials: Applications and Standards*, Upper Saddle River, NJ: Prentice Hall, 2000.

[2] N.Haller, C. Metz, P. Nesser, M. Straw, “A One-Time Password System”, *RFC2289*, February 1998.

[3] A. Jones, “Password Authentication with insecure communication”, *ACM communications*, volume 24, number 11, November 1981.

[4] A Joy and B Soh, ``A proposed secure TCP connection-oriented model for e-commerce systems,” Proceedings of International Conference on Internet and Multimedia Systems and Applications, Hawaii, Aug 12-14, pp 68-73, 2002.

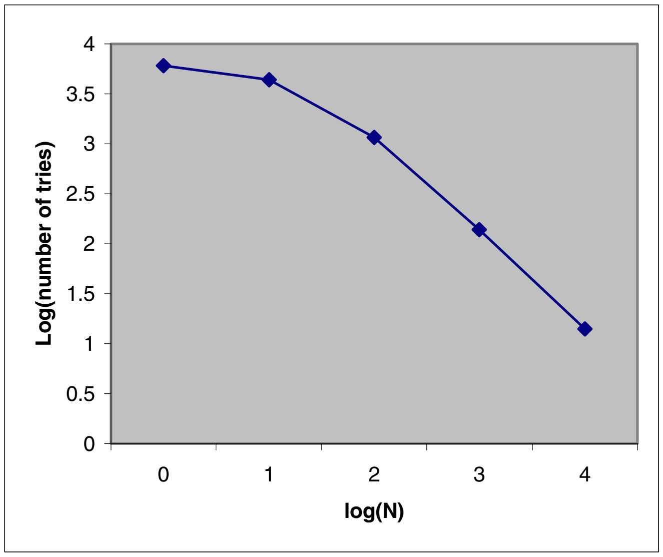
[5] National Institute Of Standards and Technology,"FIPS12 – Password Usage 1995", September 2001 http://www.itl.nist.gov/fipspubs/fip112.htm

[6] Ken Slater, *Information security in financial services*, Macmillan publishers LTD, 1991.

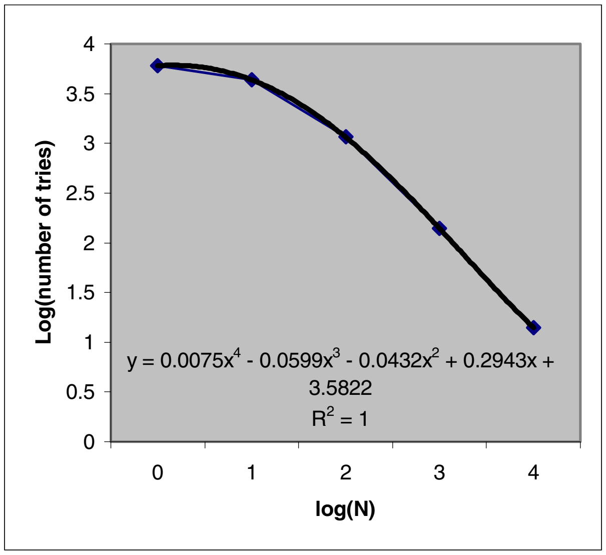
[7] N. Haller, "The S/key One-Time Password System", *RFC1760*, February 1995..

N	**Average Tries per sec**	**$\log_{10}$N**	**$\log_{10}$(average # of tries / sec)**
1	6037.844081	0	3.780881894
10	4345.922133	1	3.638081941
100	1156.140785	2	3.063010722
1000	139.3373447	3	2.14406753
10000	14.08912693	4	1.148884082

Table 1 Average number of tries per sec



Graph 1 $\log_{10}$ (N) vs. Log_{10} (Average number of tries)



Graph 2 Best-Fit: $\log_{10}$ (N) vs. $\log_{10}$ (Average number of tries)