CHALLENGE-RESPONSE AUTHENTICATION

IK2206 – Internet Security and Privacy (Group 8)

Alexander Bea
abea@kth.se

Christoph Moser
chmo@kth.se

Thomas Galliker
galliker@kth.se

Abstract
Challenge-response is a procedure with the basic task of proving the truth of information. The mechanism of challenge-response is often used to authenticate identities over unsecure networks. This report aims to illustrate different approaches of how the principle of challenge-response works, what the strengths and weaknesses are and how the risk of a successful attack against an authentication process can be diminished. Besides identity checking, the principle of challenge-response is also used in spam-filtering applications as well as in special cases of lie detection. The main finding of this report is that dictionary attacks are the only serious way of attacking a challenge-response authentication. The risk of a successful dictionary attack can be reduced by using additional salt in the hash.

Keywords: *Challenge, Response, Authentication, Spam Filtering, Lie Detection, CAPTCHA, CHAP, Java.*

1 Introduction

Ever since the means emerged to communicate through networks, security protocols have had a growing and increasingly important part to play. Private information such as passwords are still today being sent out as plaintext through the internet since secure authentication is regarded as difficult to do and thus not always used. However since there are numerous hackers out there, there are also a wide range of automated password sniffers in use on routers and other devices that connect openly to the internet, and misuse the information that they obtain. Thereby requirements that establish trust between participants on a network is more important in today's booming IT culture than ever before. Moreover how can we know that the persons we are sending important and private information to are really who they say they are? This can only be done through providing an authentication factor, which is form of evidence that proves the user's identity.

In this paper we will present and discuss one type of authentication called challenge-response authentication whereby one party presents a challenge (e.g. series of questions) and the other party is expected to provide valid answers/responses that will be authenticated. Furthermore we will cover the difference between symmetric, asymmetric and message digest based challenge-response authentication.

Thereby we will also try to establish its strengths and weaknesses and provide protocol- and implementation examples using this type of authentication. Our goal is to describe the authentication variant, but also to try answering questions like "Is the challenge-response algorithm secure?" and "What techniques can be used to strengthen security?".

2 The Principle of Challenge/Response

Challenge-response is a method which has the purpose to identify a communication partner. Figure 1 illustrates how Bob proves the authenticity of Alice using a challenge-response algorithm.

In challenge-response, one participant takes over the part of the verifier (Bob) which initialises the process by sending a challenge to the other party (Alice). A challenge is a task that could be solved only by Alice and the causer of the challenge because they own a secret [7]. If Alice can respond the expected result to Bob then Bob can be sure that he communicates with Alice. The challenge-response mechanism (as illustrated in the following sections) is a one-sided authentication. If Bob receives the expected answer from Alice then he is sure that he is talking to Alice but Alice has no evidence that Bob is her communication partner.

2.1 Identification with Symmetric Key

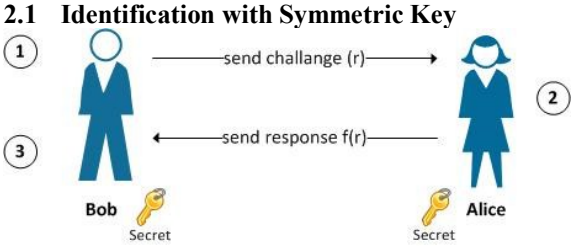

FIGURE 1: CHALLENGE-RESPONSE AUTHENTICATION WITH SYMMETRIC KEY. (DERIVED FROM [7])

If the symmetric key identification approach is used, both parties have to agree on a secret key in advanced.

1. Bob generates a random number "r" and transmits it as challenge to Alice.
2. Alice encrypts the received number with the agreed secret and sends back the result f(r).
3. Bob verifies the response by decrypting the response with the agreed secret key.

It is important that the random number is used only once. Otherwise an invader can memorise the answer and reuse it. In that case the invader could act as Alice. [7]

2.2 Identification with Message Digest

Challenge-response can make use of the irreversibility of message digest functions. The main principle remains the same while some properties change. If the message digest approach is used, both parties have to agree on a secret (e.g. a password) in advance. [6]

1. Bob generates a random number "r" and transmits it as challenge to Alice.
2. Alice uses the secret and the received challenge to a message digest function (e.g. MD5). Usually some additional salt is used to make the response more resistant against attacks.
3. Bob can verify the response by calculating the same message digest and comparing it with Alice's response.

2.3 Identification with Asymmetric Key

The challenge-response authentication could also be realized with asymmetric keys.

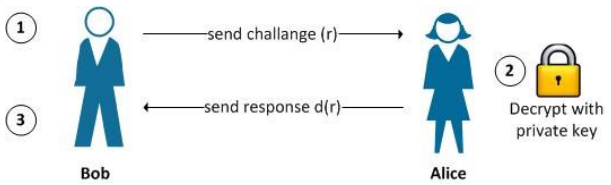

FIGURE 2: CHALLENGE-RESPONSE AUTHENTICATION WITH ASYMMETRIC KEY. (DERIVED FROM [7])

1. Bob generates a random number "r" and transmits the random number as challenge to Alice.
2. Alice decrypts the received number with her private key and sends the result back to Bob. This operation authenticates Alice, since only she knows the (private) key to perform this operation.
3. Bob can verify the response by encrypting the response with Alice's public key.

3 Strengths and Weaknesses

The following are the general strength and weaknesses of challenge-response authentication and not of any particular protocols. If a particular protocol is to be considered there might be more or less strengths and weaknesses to list. [1]

3.1 Strengths

- If a nonce and trusted intermediate is used, a secure link can be established between sender and receiver, which would protect against attacks (e.g. playback).
- The receiver can by the use of CRA provide evidence of his/her identity.
- Passwords are never sent out in plaintext, but hashed.
- The data that is to be sent can be relatively fast transmitted since it does not have to be encrypted.
- Works well and simple for spam-filtering (white- and blacklists).

3.2 Weaknesses

- Hackers can try to get the password hash of a user and create their own challenge-response by seeming as the real user.
- If the sender transmits the same challenge more than once a hacker could eavesdrop on the receivers hash from the first authentication and send it him-/her-self.
- The risks of the mentioned attacks and encryption tricking, man-in-the-middle and/or reflection attacks can be diminished through the use of e.g. a nonce and/or trusted intermediaries (such as Certificate Authorities or Key Distribution Centers).
- If only the sending party poses a challenge, there is no mutual authentication.

4 Fields of Application

As described in chapter 2, challenge-response is mainly used to authenticate people. In chapter 2 it is mentioned that C/R is only a one-sided authentication, of course, it is possible to extend the challenge-response authentication so that both communication partners can identify each other. In that case, both parties have to send a challenge to each other. This is also called mutual authentication.

The challenge-response principle is also used in some other fields of application, such as spam filtering, CAPTCHA and lie detection as well as for fake-proofs.

4.1 Spam Filtering

The main idea behind a challenge-response e-mail system is that spammers will not take the time to acknowledge the transmitted email message [9]. The email system contains two lists of addresses: a blacklist and a whitelist [9]. If a new email arrives containing a sender which is listed in the blacklist, then the email is blocked. On the other side, incoming emails with whitelisted sender addresses are always delivered to the recipient. If an email is received where the sender is neither in the black- or whitelist, a challenge is sent to the sender and the received message is queued temporarily [9]. The message can be delivered to the recipient if the sender answers the challenge correctly. In addition the sender is added to the whitelist so that prospective messages can be delivered directly.

4.2 CAPTCHA

CAPTCHA stands for "Completely Automated Public Turing test to tell Computer and Humans Apart" and it is used to ensure that for example an online form is filled in by a human [10].

It protects webpages, especially online forms, against bots which try to fill in forms automatically. A CAPTCHA has to fulfil the following characteristics: A computer must be incapable of reading the CAPTCHA using OCR techniques but a human must still be able to perceive the text.

CAPTCHA is usually used where a high risk of bot-controlled website misuse is expected. Figure 3 illustrates a CAPTCHA form where a user has to write off letters and digits from a distorted image that is part of a webpage [10]**Fehler! Verweisquelle konnte nicht gefunden werden.**.

FIGURE 3: CAPTCHA IS DIFFICULT TO READ BY A COMPUTER BECAUSE OF THE ANGLED LINE. (SOURCE: [10])

4.3 Lie Detection / Fake-Proof

The Challenge-response principle could also be used to detect faked or manipulated data. As example it is planned to use the challenge-response method for the new European road pricing system. Security and privacy are essential points which have to be considered during the development of a new road pricing system. The provider has to ensure that no location information fall into wrong hands. Therefore, location points are only stored locally on the on-board unit (OBU) of each car. The OBU transmits only the distance and the category of the road to the provider in order to create the invoice. Using the challenge-response principle, the provider is able to detect manipulated data. The provider uses alternative sources to get information about the location of vehicles, e.g. roadside-mounted radar traps, national borders and number plate scanners. If the provider has specific evidence about a car that was at a particular area at a specific time, then the provider requests the OBU to release the location of this specific point of time. Thereby, the provider is able to verify if the OBU contains genuine locations. In case of a mismatch between the OBU data and the provider data, further investigations are needed to reveal a possible forgery. [11]

5 Challenge-Handshake Authentication Protocol

The Challenge-Handshake Authentication Protocol (CHAP) is a concrete implementation of the challenge-response algorithm. It is ratified in RFC1994 [997]. CHAP is a fundamental part of the Point-to-Point Protocol (PPP), a data link protocol to establish private connections between Internet nodes. PPP belongs to the Internet protocol suite.

Although Microsoft's proprietary versions of CHAP, MS-CHAPv1 and MS-CHAPv2, enjoyed greater publicity and broader acceptance, they are nowadays considered as insecure [2], [3]. MS-CHAPv2 provides mutual authentication between two peers. This is basically done the same way as the one-way authentication, only that both, the client and the server, act as challenger.

MS-CHAP reached its popularity with the commercialisation of WLAN (IEEE802.11). The Extensible Authentication Protocol (EAP) is an authentication framework that allows several authentication protocols to be used. Many authentication protocols used with EAP are based upon the ideas of challenge-response authentication (e.g. EAP-PSK, LEAP).

6 Challenge-Response in Java

As already stated in the explanation of the challenge-response algorithm, an authentication process has to pass through three simple steps. In this chapter the relevant steps to perform a Message Digest-based challenge/response authentication between two

communication partners are roughly explained and supported by simplified code snippets.

- Step 1: Generate Challenge

A client (let's call it "Alice") requests from her authenticator ("Bob") a challenge. A challenge is nothing else than a randomly generated number[1]. Our sample code makes use of the SecureRandom class to generate a 1024 bit number.

```
SecureRandom sr =
SecureRandom.getInstance("SHA1PRNG");

byte[] bytes = new byte[1024 / 8];
return sr.nextBytes(bytes);
```

The issuer of the challenge, "Bob", does two things with the newly generated challenge: First he stores it to a local variable and then he returns it to the requester, Alice.

- Step 2: Calculate Response

As soon as the requester, Alice, gets the challenge, she calculates a response code. This step is crucial for the security of the algorithm and must be implemented carefully to not allow attackers to misuse possible weaknesses.

The challenge code that we got from the authenticator is concatenated with Alice's username and password and then passed to a message digest function. In this case we use MD5 as hashing algorithm. The output of hash functions is an irreversible sequence of bytes. The more components are used to calculate the response, the more secure is the authentication. Using time stamps as additional "salt" of the hash input makes the response more resistant against dictionary attacks but expects both involved systems to have roughly synchronised clocks.

```
MessageDigest md =
MessageDigest.getInstance("MD5");

md.update((username + password + challenge +
timestamp).getBytes());

return md.digest();
```

Alice sends the response back to Bob if the calculation is done.

[1] It is actually a pseudo-random number (PRN) since the generation of truly random numbers is hard to achieve.

- Step 3: Validate Response

Bob is now in charge to check the correctness of Alice's response. Bob has all information to calculate the response and checks whether it matches: The challenge (since Bob saved it in a variable), the username and password (since this information is stored in a database). If Alice's and Bob's calculated response match, the authentication was successful.

```
if (Utils.generateResponse(username,
password,
challenge).equals(responseFromClient)) {
     //Login successful
} else {
     //Login failed
}
```

If no random challenge was used, an attacker could listen for hashes of username/password combinations on the network and use them to illicitly gain access.

7 Conclusion

- Challenge-response algorithms provide an authentication mechanism that is protected against playback attacks since they use random challenges.
- The principle of challenge-response can be used in various other applications such as truth tests, spam filtering, etc.
- The only serious way of attacking challenge-response based authentication is to run a dictionary attack.
- Strong passwords and salting of the hash helps diminishing the success of a dictionary attack.
- The challenge (variable "r" in the examples of chapter 2) must not be predictable in any way. Long random strings are most appropriate.
- To mitigate the success likelihood of online dictionary attacks as well as brute force attacks, it is important that authentication systems implement timeouts between unsuccessful login attempts.

8 References

[1] P. Sjödin, “Authentication Protocols and Key Establishment”, Kungliga Tekniska högskolan, 2011

[2] B. Schneier, “Mudge, and D. Wagner, Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)”, http://www.schneier.com/paper-pptpv2.pdf, 1999, *[last access: 24.11.2011]*

[3] J. Eisinger, “Exploiting known security holes in Microsoft's PPTP Authentication Extensions (MS-CHAPv2)”, http://penguin-breeder.org/pptp/download/pptp_mschapv2.pdf, 2001, *[last access: 24.11.2011]*

[4] W. Simpson, “PPP Challenge Handshake Authentication Protocol (CHAP)”, http://tools.ietf.org/html/rfc1994, 1996, *[last access: 24.11.2011]*

[5] F. Bersani, H. Tschofenig, “The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method”, http://tools.ietf.org/html/rfc4764, 2007, *[last access: 24.11.2011]*

[6] R. Hafernik, “Challenge-Response Authentication in Java”, http://roborant42.appspot.com/show/entry/4025, 2007, *[last access: 24.11.2011]*

[7] J. B. Knudsen, “Java Cryptography, Chapter 6: Authentication”, http://oreilly.com/catalog/javacrypt/chapter/ch06.html, 1998, *[last access: 24.11.2011]*

[8] J. Swoboda, S. Spitz, M. Pramateftakis, “Kryptographie und IT-Sicherheit, Grundlagen und Anwendung“, Vieweg+Teubner Verlag, 2008

[9] BareMetal.com Inc, “Introduction to our Challenge-Response e-mail system”, http://domain-dns.com/docs/challenge_response.html, 2003, *[last access: 24.11.2011]*

[10] Carnegie Mellon University, “CAPTCHA: Telling Humans and Computers Apart automatically”, http://www.captcha.net, 2010, *[last access: 24.11.2011]*

[11] M. Pouly, “Billing Systems”, University of Luxembourg, 2011