Location-Based Kerberos Authentication Protocol

Abdelmajid, N.T., Hossain M.A.  
School of Informatics  
University of Bradford, UK  
{ntjabel, mahossain1}@brad.ac.uk

Shepherd, S.  
School of Engineering  
University of Bradford, UK  
sjshepherd@brad.ac.uk

Mahmoud, K.  
Department of Computer Engineering  
Zarqa University, Jordan  
k.w.mahmoud@zpu.edu.jo

Abstract--Online communication offers organizations greater efficiency. However, online processes increase the threat level during message transfer. This necessitates researchers to develop and improve security protocols in order to enhance the security of communication lines. Despite the evaluation and acceptance of many authentication protocols, online communications remain insecure. We propose to add the user’s physical location as a new authentication factor into Kerberos protocol and call it N-Kerberos protocol, and we validate the new form of Kerberos (N-Kerberos).

Keywords—GPS; Kerberos Protocol; N-Kerberos protocol

I. Introduction

Most organizations use online communication rather than more traditional forms. However, online systems provide an opportunity for hackers and thieves to carry out their malicious works. To overcome this problem, security specialists have devised many encryption codes, such as RSA, DES or MD5, to create a secure channel for clients to communicate. Although RSA is one of the most powerful encryption codes, certain security protocols that use RSA remain open to attack [1, 2, 3, 4, 5]. Therefore, most people do not feel that online communication is a secure environment. Kerberos is one of the most common key distribution protocols [8]. Even though Lowe [9] concluded that Kerberos is the strongest form of authentication in his hierarchy of authentication specifications, Kasslin and Tikkanen [10] proved that in some cases Kerberos suffers from replay attack.

This paper presents our improvement to Kerberos protocol that we call (N-Kerberos) by adding user's position co-ordinates as a new factor in the authentication process. We have examined two different situations of this new technology. In the first case, the server receives a signal from the user indicates his physical location, and the server needs to verify the validity of user's physical location. While in the second case, the server sends a signal to the user, and the user compares it with what he has to substantiate his physical position.

The remainder of this paper is set up as follows; Kerberos protocols are presented in two. Section three talks about the weaknesses and the limitations in Kerberos. In section four, we introduce the modification of Kerberos protocol. In section five, the evolution of technology from the first case of the second case has been shown, and views the results of the examination. Finally, our conclusion and future work are discussed in the last section.

II. Kerberos

In order workstation benefits from the services provided by servers, they are required to be authenticated. need to access servers to complete the processes, they are required to be authenticated. Kerberos is designed to authenticate the end-user to the server. To understand how Kerberos works, we divide it into three different steps:

A. Authentication exchange:
- The client requests a ticket from the authentication server (AS) to interact with the ticket-granting server (TGS) as shown in figure3 (KRB AS REQ).
- AS then checks up the client in its database and generates a session key (SK1) to use between the client and the TGS (SK1C:TGS).
- Kerberos encrypts the SK1 using the client’s secret key. The AS also uses the TGS’s secret key (KAS:TGS) to create and send the user a ticket-granting ticket (TGT). It is shown as (KRB AS REP) in figure3.

B. Ticket-Granting Service exchange:
- The client decrypts the message and recovers the session key, then uses it to create an authenticator containing the user’s name and a time stamp.
The client then sends this authenticator, along with the TGT, to the TGS, requesting access to the target server (KRB_TGS_REQ).

The TGS decrypts the TGT, and then uses the SK1 inside the TGT to encrypt the authenticator. It verifies information in the authenticator, the ticket and the time stamp. If all match then it allows the request to proceed.

Then the TGS creates a new session key (SK2) for the client and application server (AP) to use, then encrypts it using SK1 and sends it to the client.

The TGS also sends a new ticket containing the client’s name, a time stamp and an expiration time for the ticket (KRB_TGS_REP), all encrypted with the AP’s secret key (K_{TGS-AP}).

C. Client/server exchange:

- The client decrypts the message and gets the SK2.
- Finally ready to approach the AP, the client creates a new authenticator encrypted with SK2.
- The client sends the session ticket (already encrypted with the AP’s secret key) and the encrypted authenticator. Since the authenticator contains plain text encrypted with SK2, this proves how the client knows the key (KRB_AP_REQ).
- The AP decrypts and checks the ticket, the authenticator and the time stamp.
- For applications that require two-way authentication, the AP returns a message consisting of the time stamp plus 1, encrypted with SK2. This proves to the client that the server actually knew its own secret key and thus could decrypt the ticket and the authenticator.

Figure 1 shows the contents of Kerberos messages. We refer to AS and TGS as Key Distribution Center (KDC). Figure 2 shows the Kerberos messages. Message 1 is sent from client A to KDC (given the symbol S) requesting a ticket to use a service from AP (given the symbol B). Then, S sends A message 2. Message 2 includes the ticket ((N_s, K_{ab}, A_jk_{ts}). Detailed analysis of Kerberos is available in [12, 13, 14, 15].

1: A \rightarrow S : A, B
2: S \rightarrow A : \{N_a, B, K_{ab}, \{N_s, K_{ab}, A\}_{K_{ts}}\}
3: A \rightarrow B : \{N_s, A, K_{ab}\}_{K_{ts}}, \{N_a, A\}_{K_{ab}}
4: B \rightarrow A : \{N_b + 1\}_{K_{ab}}

Figure 2. Kerberos protocol

Although Kerberos has a full BAN guarantee [8] and is trusted by many authors [16, 17], a number of weaknesses have been found in its messages by Bellovin et al [18]. This shows that Kerberos needs further investigation. In the next section, we discuss a number of weaknesses and limitations in Kerberos.

III. Weaknesses in Kerberos

There are a number of weaknesses apparent in Kerberos. Davis and Swick [19] show some of Kerberos’s problems. In this section, we demonstrate a replay attack problem. A replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. Kerberos has many mechanisms aimed at making replay attacks difficult. One of these mechanisms is that authenticators rely on machines’ clocks being roughly synchronized. We are not confident with this point for several reasons; in some cases, synchronization protocols are unauthenticated [20, 21]. In addition, when a server is misled about the correct time, attackers can easily replay the authenticator. Moreover, the attacker might be able to mount the attack within the configured time. Furthermore, if the time is configured to be too small, the client will face problems in time synchronization.

Using a cache memory to store used authenticators is another mechanism to guard against reuse. This cache should hold all authenticators used within the allowable time skew. If the server uses a cache for the used authenticators, a passive attack becomes impossible. A server will reject all authenticators it has already seen [23]. Authenticator caching makes replay attacks slightly more difficult, but it is not a sufficient protective mechanism; keys have to be saved in particular storage area. Kerberos modification was made to store keys in
shared memory. However, this area could be attacked. The authenticator key returned by TGS is stored in an accessible area. Therefore, the intruder can crack the local computer protection and steal the key. On the other hand, performing cache mechanisms is not a good idea for some other systems. For example, it is very difficult for TCP-based servers to store authenticators in UNIX system [24]. Moreover, it is easy to store the authenticator in UDP-based servers. However, the problem lies when a client retransmit a request in the case that the server's response was lost. [25].

The third Kerberos mechanism to stop replay attack is that, the ticket inside KRB_AP_REQ should include the network address of the client. It should verify that the source address of the message matches the address in the ticket. Again, this mechanism is insufficient. In some cases, KDC will not include network addresses in the tickets it gives out at all. Even if the ticket did include a network address, it is noted that the network address is under full control of the attacker [26].

Many protocols in Microsoft Windows domain use Kerberos v5 as the primary authentication mechanism. SMB (Server Message Block) and LDAPv3 (Lightweight Directory Access Protocol) are examples of such protocols. SMB and LDAPv3 protocols may be attacked by replay attack, password attack against TGT or pre-authentication data, and attack against message delivery time. Through a replay attack, the attacker will be able to access the shared files and modify directory entries with the victim's credentials. A Replay Attack on Kerberos v5 exploits the final message, KRB_AP_REQ. If an attacker is able to access the network traffic from the victim, he will be able to extract the KRB_AP_REQ sent by the victim, he can then simply attempts to re-use this message to authenticate himself to a server. In some cases, the server will accept the replayed message sent by the attacker allowing him full access to the service with the victim's credentials [10].

We have to say that we are not suggesting that Kerberos is ineffective. The effectiveness of Kerberos relies on the server configurations and implementations. Some implementations aren't performed in a correct way, while others have a default configuration, which might make them susceptible to a replay attack. Therefore, to stop replay attacks, the server needs to be configured correctly by using integrity protection. However, such implementation is not easy and is not without its costs. We should point to the large size of the verifications that are used in Kerberos protocol. The proliferation of these definitions and complexity increase the likelihood of errors. The required time period is dependent on the service. For example, the time required to complete a particular user request may differ from that required for another different request, thus making it difficult to define the time accurately. This leads to the possibility of hackers exploiting this weakness. Denning and MacDoran [27] show a detailed explanation of these definitions. The new challenge in this study is to protect the user, even in the case of failing to implement the necessary definitions in the Kerberos system. To achieve this, we propose a new form of Kerberos that we call N-Kerberos. In the next section we give details of N-Kerberos protocol.

IV. N-Kerberos: Location based Kerberos

Although Kerberos is one of the most secure protection key exchange mechanisms, it is not strong enough. Our main concern is to eliminate the possibility of replay attack. Inserting a network address into a ticket will not add any extra security [26]. In fact, it is to stop any other users of immediate reuse from a different network. We propose a modification on Kerberos that we call N-Kerberos. It works by adding client's physical position address. This Address can be determined by a Global-Position-System (GPS) receiver. The great development that has been achieved on the GPS over the past few years gave an indication of a potential for integration with different techniques to raise the level of data protection. GPS however is used only outdoors in the sense that the receiver should have a direct "view" to at least four GPS satellites. It is not easy for any entity in cyberspace to pretend to be in any place other than where its LSS actually is [33]. Therefore, cracking user's position is very complicated [34].

We have proposed to add the user’s physical address to all messages given out by Kerberos, in addition to the previous two conditions; encrypting the messages using a strong password and having a time stamp. Our proposed work requires the server to have a database of a list of legitimate users' positions addresses. Having these addresses will enable the server to test out the availability of a user's position address before allowing the users to utilize the services. We have studied this idea in three different methods, stated as follows:

Method1: in this method, the task of confirming the authenticity of the location falls into the responsibility of the server. Figure 3 shows the messages of method1 of N-Kerberos protocol.

In the first message, the user A sends a request to obtain the key, which can be used to communicate with the user B. Then, the server sends his response to the user A. This response contains a special ticket (\( \{ N_s, GPS_a, K_{ab}, A \} K_{is} \)) to be sent by A to B through message 3. Note that it will be preceded by the implementation of many of the Kerberos procedures, noted in reference [27].
1. \( A \rightarrow S : A, N_a, B \)
2. \( S \rightarrow A : \{ N_s, GPS_{a,b}, B, K_{ab}, \{ N_r, GPS_{a,b}, K_{ab}, A \} \}_{K_{sa}} \)
3. \( A \rightarrow B : \{ N_s, GPS_{a,b}, K_{ab}, A \}_{K_{sa}}, \{ A, GPS_{a,b}, N_a \} \)
4. \( B \rightarrow A : \{ GPS_{b}, N_a + 1 \} \)

Figure 3. Method 1 of N-Kerberos protocol

We add new procedures for the use of the GPS, which are as follows:

1) The server uses the list of legitimate users' physical location addresses to add A's physical address to the ticket (\( \{ N_s, GPS_{a,b}, K_{ab}, A \} \))
2) The physical location address of B must be added to the message (\( \{ N_s, GPS_{a,b}, B, K_{ab} \} \))
3) Message 3 has both a ticket and authenticator. The ticket (\( \{ N_s, GPS_{a,b}, K_{ab}, A \} \)) is encrypted by \( K_{sa} \) and the session key \( K_{ab} \) encrypts the authenticator (\( \{ A, GPS_{a,b}, N_a \} \)).

The following are modifications to the previous Kerberos protocol.

1) A adds his physical location address derived from the GPS receiver to a part of the authenticator.
2) B will not believe the message unless the GPS_{a,b} located in part of the ticket, sent by server, matches the GPS_{a,b} located in part of the authenticator, sent by A.

In order for A believe B, both the second message and the fourth message will be used to validate B's physical location. To accomplish this, the second message will serve as the ticket, and the fourth message will serve as the authenticator. The GPS_{a,b} located in the fourth message, derived from the GPS receiver of B, must match the GPS_{a,b} located in the second message, sent by the server.

We proposed this method to force the users to use their pre-defined physical location addresses, stored in the server, in order to acquire the private key which can be used to communicate with each other.

Unfortunately, a problem was identified during the examination of this method. The problem is in the second message; it does not contain the physical address of user A, where there is no evidence that the recipient of the message 2 is the user A. Consequently, the hacker can steal both the second and fourth messages and perform the required comparison of the \((GPS_{a,b})\) in message 2 with the \((GPS_{a,b})\) in message 4 without having to confirm his physical location. This means that the addition of the GPS feature in the second message did not add to the security of the message. To solve this, we propose method 2 as follows.

**Method 2:** In method 2, the responsibility of confirming the authenticity of the location falls to the user instead of the server, as it was in method 1. Figure 4 shows method 2 of N-Kerberos protocol.

1. \( A \rightarrow S : A, N_a, B \)
2. \( S \rightarrow A : \{ N_s, GPS_{a,b}, B, K_{ab}, \{ N_r, GPS_{a,b}, K_{ab}, A \} \}_{K_{sa}} \)
3. \( A \rightarrow B : \{ N_s, GPS_{a,b}, K_{ab}, A \}_{K_{sa}}, \{ A, N_a \} \)
4. \( B \rightarrow A : \{ N_a + 1 \} \)

Figure 4. Method 2 of N-Kerberos protocol

The method used in this method 2 is quite different from the previous model. These differences are shown as follows.

1) The second message contains the physical address of user A \((GPS_{a,b})\) instead of the physical address of the user B \((GPS_{b})\). A has to prove that he is using the legitimate position. This can be achieved by comparing the physical address in the message, sent by the server \( \{ N_s, GPS_{a,b}, B, K_{ab} \} \), with that acquired from the GPS receiver, which is installed in A's location. Thus, A will not be able to obtain the key \( K_{ab} \) in the absence of matching addresses.
2) The ticket included in the second message \( \{ N_s, GPS_{a,b}, K_{ab}, A \} \), sent from the server, contains the physical address of user B \((GPS_{b})\) instead of the physical address of user A \((GPS_{a,b})\). B has to prove that he is using his legitimate position. This can be achieved by comparing the physical address in the ticket, sent by server \( \{ N_s, GPS_{a,b}, K_{ab}, A \} \), with what would receive it from his GPS receiver which is installed in his location. Again, B will not be able to get the key \( K_{ab} \) in the absence of matching addresses.
3) There is no need to add the GPS_{a,b} in both the authenticator and message parts of message 3, and no need to include GPS_{b} in message four, as in this method, there is no longer a need to compare the GPS_{a,b}'s in message three, and the GPS_{b}'s in messages two and four.

The Client's physical location must be used for the key exchange of the communication. Subsequently, the key can be used from any other place. Clearly, this causes a limitation, however this modification provides more protection against replay attacks. Unfortunately, another problem was found in this method; there is nothing forcing the user to make a comparison between the two physical addresses. In other words, there is nothing...
prevent a hacker from stopping the comparison process, or to make it appear as if the comparison result is positive. As a result, method 3 was introduced to force users to make the comparison.

**Method 3:** To understand what we did in this method, a clarification of the two components of the GPS signal is required. Each GPS satellite transmits two types of signals, a secure encrypted signal exclusively for military users, encrypted by P(Y) code, and a non secure civilian signal, encrypted by coarse/acquisition code (C/A). The second type is available to the public and is extremely vulnerable to electronic attacks [28]. The P(Y) code needs special hardware, available to the U.S. government, to be decrypted, and is design to resist electronic attacks. A beneficial property of the P(Y) code is that it can be used to figure out the physical position. On the other hand, it is quite difficult to know what the P(Y) code is using the physical location. This means that users can capture their P(Y) code and store it to verify their physical address signature to the server. In method 3, we propose using this method to that effect. Figure 5 show method 3 of N-Kerberos protocol.

1. A → S : A, N_a, B
2. S → A : {N_s, Sig_a, P, B, {N_s, Sig_b, P, A}K_a}K_a
3. A → B : {N_s, Sig_b, P, A}K_a, {A, N_a}K_ab
4. B → A : {N_a + 1}K_ab

![Figure 5. Method3 of N-Kerberos protocol](image)

As shown in figure 5, we hash the signature P(Y) code and uses the value of hashing in encrypting the key K_ab. This will enforce the user to decrypt the signature using his P(Y). User B has to do the same to read the key from the ticket when received through message 3. The modifications to method 2 are as follows.

1. The server needs to capture the P(Y) of all users.
2. The server needs to hash the P(Y) code and encrypt the Key (K_ab) using the hashed value of the P(Y) code of A. Sig_a is the encrypted form of the key.
3. User A needs to decrypt the message by the key K_ab, and decrypt the key K_ab using the hashed value of his P(Y) code.
4. The ticket, which is sent through message 3 has B’s signature. B needs to decrypt the ticket using K_ab, and then to uses the hashed value of his P(Y) code to decrypt the K_ab.

By this technique, we enforce the users to use their physical location addresses signature in order to read the key. The attacker will be forced to use a maximum amount of time trying to decrypt Sig_a, which will cause problem with time synchronization. In the next section we implement method 3 of N-Kerberos.

VI. Testing

The second message of Kerberos has been selected for testing. Some pre-actions have to be performed before starting the test, which are illustrated in the following list:

1. To prepare a GPS receiver.
2. To capture the P(Y) code of A and B.
3. To create two different signatures; the hashed value of A’s P(Y) code, and the hashed value of B’s P(Y) code.
4. To save the encrypted signatures in the server’s database.
5. To prepare a tool to steal and decrypt the packet.
6. We have assumed a weakness in the setting of prior verification to the use of Kerberos such as, using a weak password for packet encryption and the allowed period time for sending the message is not important. Therefore, the message does not have a specific expired time.

Testing was performed and is described in the following cases:

**Case 1:** in this case, we used Kerberos protocol using weakness verification and perform the following steps:

1. Send an encrypted packet to A, using weak password and with open period of time.
2. Steal the packet and decrypt it.

Following testing, we broke the code and decrypt the message, because of the use of a weak password and not using a determination of time.

**Case 2:** In this case, we used N-Kerberos protocol using the same weaknesses definitions used in the first case. It is essential for hackers to penetrate different levels of protection. The first level is to breakthrough the encryption mechanism as described in the first case. The second level is to penetrate the GPS protection, as shown in the following figure.

![Figure 6. different layer of protection](image)

We repeat steps in method 1 and we success to the penetration of the first level of protection (K_ab). Despite this, we were not able to read the K_ab. To do so, the second level of protection (GPS) must be breached. Unfortunately, Sig_a could not be decoded.

We believe now, the P(Y) code can be used to make the replay attack more difficult, even in the case of bad verification of Kerberos protocol.
VII. Conclusion and Future Work

The main challenge in this paper is increasing Kerberos protocol’s ability to protect against replay attacks. We emphasize that the problem lies in that it is not simple for users to implement Kerberos’s predefinitions. This might be due to users not following the correct procedures to implement these protocols. In this paper, we make it imperative for the user to be more protected. To do this, users are forced to use a very secure location signature created using the $P(Y)$ code. We proposed adding the user's physical position to Kerberos protocol's messages as a new factor, in addition to the obtained factors, which reduces the possibility of replay attacks occurring. We then validated the proposed protocol using different methods. Our findings show that in order for this to work, the server needs to have a list of the physical addresses of the legitimate participants. The overall modification of our study is adding a new layer of protection using a user’s physical location.

For our future work, we propose that the $P(Y)$ code signature is injected into the user’s device instead of having to carry the GPS receiver all the time.

References


http://www.insidegnss.com/auto/sepoct09-Lo.pdf"