Implementation of Crossrealm Referral Handling in the MIT Kerberos Client

Jonathan Trostle, Irina Kosinovsky
Cisco Systems
170 W. Tasman Dr.
San Jose, CA 95134

Michael M. Swift
University of Washington
Dept. of Computer Science and Eng.
Seattle, WA 98105

Abstract

The Windows 2000 Kerberos implementation [1, 2] uses a different approach to solve the Kerberos realm resolution problem than has traditionally been used by MIT Kerberos implementations. In this paper, we present the details of the two approaches and compare them. To facilitate more effective use of the Kerberos ticket cache, we propose a new format for referral data that includes a list of alias names as part of the returned referral information. We include the pseudocode for the algorithm that we have implemented in the MIT Kerberos client that allows a MIT Kerberos client to request and follow referrals from a Windows 2000 Kerberos KDC, thus removing the need for management and administration of DNS to realm mapping files on Kerberos client hosts. We conclude with a discussion of issues that are applicable to any mutual authentication protocol.

1 Introduction

The Kerberos protocol [2, 4] allows security principals (network entities such as users or servers) to be separated into realms, allowing for separate administrative control. The protocol still allows authentication between principals in different realms using *crossrealm* authentication, in which case a user from one realm can mutually authenticate with a service in another realm. The MIT Kerberos implementation of crossrealm authentication depends on clients being configured with information about multiple realms. The Windows 2000 Kerberos implementation introduced a new mechanism for migrating this information to the Key Distribution Centers (KDC's), removing the need to separately administer clients. This paper presents the specification and design approach for the integration of the Windows 2000 Kerberos crossrealm referral algorithm into the MIT Kerberos client on the Solaris operating system. We have successfully tested our implementation against the Windows 2000 KDC for crossrealm operations in a Windows 2000 domain (realm) hierarchy.

Kerberos is an authentication and key distribution system based on the Needham-Schroeder protocol [5] and developed at MIT. In the protocol, a central server, called the Key Distribution Center, stores keys for users and servers. It uses a ticket mechanism to let clients and servers mutually authenticate each other. To enable scalability beyond a single set of trusted administrators, Kerberos incorporates the notion of *realms*, which are separate administrative entities. Within each realm, a single administrator has complete control over authentication and the set of users and services which may authenticate. To allow realms to cooperate, they may establish a trust relationship, in which the realms share a key. One realm can then vouch for the authenticity of a user to the KDC of another realm. Furthermore, realms are named using the DNS naming convention [8], and trust relationships can be transitive (the realms can be arranged in a tree where each realm only shares a key with both its parent and its children).

The standard approach for a user wishing to authenticate to a service in another realm is to explicitly request tickets for all the realms on the trust path between the user's and the service's realms. However, this approach requires that the client have knowledge of the trust relationships between realms. The implementation of Kerberos in Windows 2000 modified this procedure by letting the KDC return referrals for other domains. In this technique, the user requests a ticket to the service directly from its own KDC, which may then return a referral to the next realm on the trust path. In this way, clients do not need to be aware of the trust relationships between realms. This paper presents the design of this referral mechanism as well as an implementation using MIT Kerberos on the Solaris operating system.

We first describe the Kerberos protocol in section 2 and describe the traditional method of crossrealm au-

thentication. In section 3, we describe the mechanism used in Windows 2000 through the use of a simple example. Section 4 provides more detail on the protocol, including specifications for the messages used. Section 5 provides a comparison of the two approaches with respect to security, performance, and ease of administration. Section 6 contains the high-level design and pseudo-code for the protocol implementation. We discuss naming issues relating to Kerberos and authentication in section 7, and conclude in section 8.

2 The Kerberos Protocol

The Kerberos [2, 4] protocol was developed at MIT to provide authentication in a distributed network of clients and services. The protocol is based on the Needham and Schroeder protocol [5], in that clients contact a central server to request tickets to services. A ticket is a data structure identifying the client and encrypted with the secret key of the server. The server trusts that only it and the KDC know this secret key, so it trusts that the KDC created the ticket. Clients obtain ticket through two protocols: the *authentication service* (AS) protocol, in which case the reply from the KDC is encrypted with the secret key of the user, and the *ticket granting service* (TGS) protocol, in which case the reply from the KDC is encrypted with a session key known to the client and KDC.

Kerberos [2, 4] principals are organized into realms. A Kerberos realm is an administrative unit; typically, common security policies such as a password policy would be enforced on a per realm basis. A Kerberos KDC stores the secret keys for all secret key Kerberos principals in its realm so that it can issue tickets to client principals in its realm. Realms can be linked by either secret key trust relationships or public key trust relationships [7]. In Kerberos version 4 [6] (and earlier versions of the Windows operating system that use the NTLM security protocol), realms can only have direct trust relationships, in that users from one realm can only authenticate to realms that share a key with their realm. Kerberos V5 improves the scalability of this approach by allowing transitive trust that results from a tree trust structure with possible additional shortcut trust relationships. Figure 1 shows a Kerberos V5 trust graph.

There are two issues that arrive when crossrealm authentication is used: determining the realm of the server, and determining the trust path to reach that realm. The first problem is to determine, given the name of a service, what realm it belongs to. This problem is compounded if the server may have many names, such as in the case of a multi-homed computer with multiple DNS names. In this case, the name may have to be *canonicalized*, or mapped to a single name. This process may need to take place before the determination of its realm can be made, and can be performed either at the client or the KDC. In other words, when a client desires to use Kerberos authentication when communicating with another Kerberos principal, it must determine the realm for that principal before sending a TGS_REQ message to a Kerberos KDC. The reason is that the server's realm is a required field in the request body of the TGS_REQ message:

```
KDC-REQ-BODY ::= SEQUENCE {
    kdc-options[0]              KDCOptions,
    realm[2]                    Realm,
    sname[3]                    PrincipalName
                                OPTIONAL,
    from[4]                     KerberosTime
                                OPTIONAL,
    till[5]                     KerberosTime
                                OPTIONAL,
    rtime[6]                    KerberosTime
                                OPTIONAL,
    nonce[7]                    INTEGER,
    etype[8]                    SEQUENCE OF
                                INTEGER,
    addresses[9]                HostAddresses
                                OPTIONAL,
    enc-authorization-data[10]  EncryptedData
                                OPTIONAL,
    additional-tickets[11]      SEQUENCE OF
                                Ticket
                                OPTIONAL
  }
```

The second problem, of how to determine the path that authentication should take through the set of realms, can also be determined at the client or at the KDC. In the first case, the client would need to know the structure of the domain hierarchy, which could be determined from the lexical structure of the realm names. For example, to traverse from the realm EXAMPLE.COM to DEV.PRODUCTS.EXAMPLE.COM, the client could assume that PRODUCTS.EXAMPLE.COM is the intermediate realm. This process is complicated, though, by the presence of short-cut trust relationships, such as when realms that aren't adjacent in the naming tree directly trust each other. In this example, the EXAMPLE.COM realm may have a direct trust relationship to DEV.PRODUCTS.EXAMPLE.COM. In this case, for clients to determine the authentication path, they would need to know about all short-cut

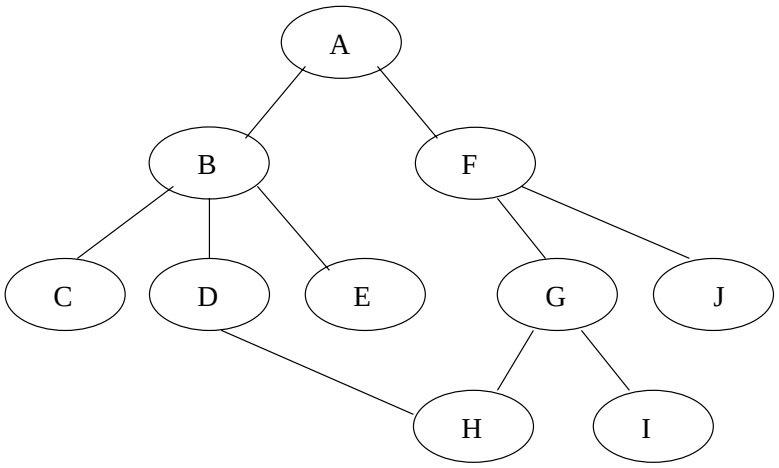


Figure 1: Sample Kerberos Crossrealm Trust Graph: A Line Between Two Vertices of the Graph Indicates a Shared Secret Key Trust Relationship Between the Corresponding Realms

trust relationships. If the KDC is responsible for determining the path, then only the KDC's need to be aware of these trusts.

The MIT implementation of Kerberos depends on the client to perform all three functions: name canonicalization, realm determination, and trust path determination. When the MIT Kerberos client creates a TGS request message to send to the KDC (in order to obtain a service ticket) it determines the realm of the server principal by looking in a configuration file. If an organization's realm structure changes, the configuration file on all MIT Kerberos clients must be manually updated. Therefore, the current MIT Kerberos client is not scaleable in an environment with multiple realms where the realm hierarchy is dynamic.

2.1 Name Terminology

Here we briefly review some terminology on names given a hypothetical host that is running a Kerberos service. The host has a canonical (FQDN) DNS name: host.example.org. The host may also have an alias DNS name: service.example.org. In this case, the host is running a Kerberized service with the service identifier "service". Thus the canonical Kerberos principal name for the service is service/host.example.org. The alias Kerberos principal name is service/service.example.org.

3 Two Approaches to the Kerberos Server Principal Name Realm Resolution Problem

The Windows 2000 implementation of Kerberos attempts to address this scalability problem by centralizing realm information at the KDC. In this case, all three operations (name canonicalization, realm determination, and trust path determination) are all performed at the KDC rather than the client. This protocol change eliminates the scalability issues present in the MIT Kerberos implementation. The crossrealm referral feature also reduces ongoing support costs, because it reduces the dependence on a direct mapping from DNS domains to Kerberos realms. The protocol extension works by allowing a client to request a ticket to any service, in any realm, from its KDC. If the service is not in the same realm as the client, then the KDC will return a *referral*, which consists of a ticket to the next realm on the trust path and a data field containing the name of the realm containing the service. The client may then, at the next realm, ask for a ticket to the destination realm. In this case, the KDC will again return a ticket to the next realm on the trust path, but omits the destination realm information. The client repeats this process until it reaches the destination realm, where it may request a ticket to the service again.

In order to better understand the standard MIT client as well as the crossrealm referral feature, we present an example.

Figure 2 shows the case with three realms: LOC.EXAMPLE.COM, REM.EXAMPLE.COM, and EXAMPLE.COM:

Here realm LOC.EXAMPLE.COM has a bidirectional trust relationship with EXAMPLE.COM, and REM.EXAMPLE.COM has a bidirectional trust relationship with EXAMPLE.COM. Suppose

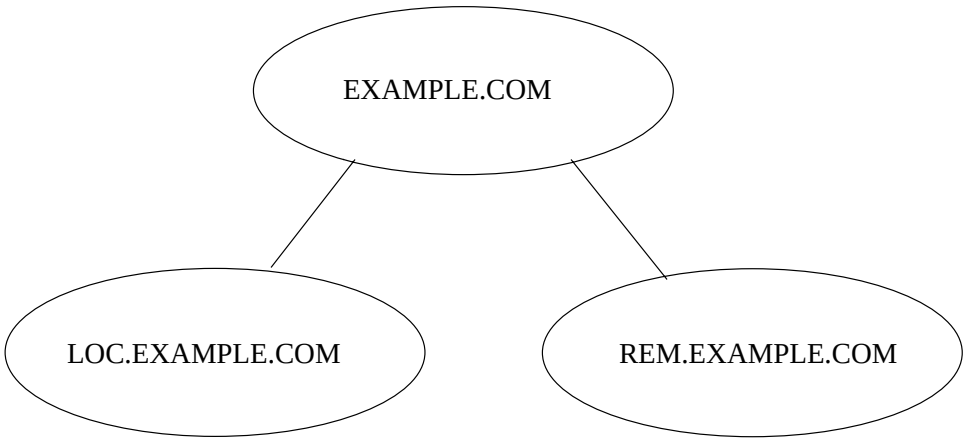


Figure 2: Simple Kerberos Realm Trust Tree

bob@LOC.EXAMPLE.COM desires to contact the server srv.r.example.com. In the MIT client case, it looks in the client configuration file. Typically, there would be an entry for r.example.com:

```
...
r.example.com REM.EXAMPLE.COM
...
```

The r.example.com entry indicates that the DNS domain r.example.com maps to the Kerberos realm REM.EXAMPLE.COM. Thus, srv.r.example.com is a Kerberos principal in the realm REM.EXAMPLE.COM. The client will obtain crossrealm tickets for EXAMPLE.COM, and REM.EXAMPLE.COM, by using its knowledge of the realm trust tree. It will then obtain a service ticket for the server srv.r.example.com (see Figure 3).

The new crossrealm extension operates as follows: bob@LOC.EXAMPLE.COM desires to contact the server srv.r.example.com. Bob's Kerberos client will send a TGS request message to the KDC in LOC.EXAMPLE.COM. The request will have the name canonicalize bit set in the KDC options field. Upon seeing the bit, the KDC will attempt to determine the realm for the server name (srv.r.example.com) in the request.

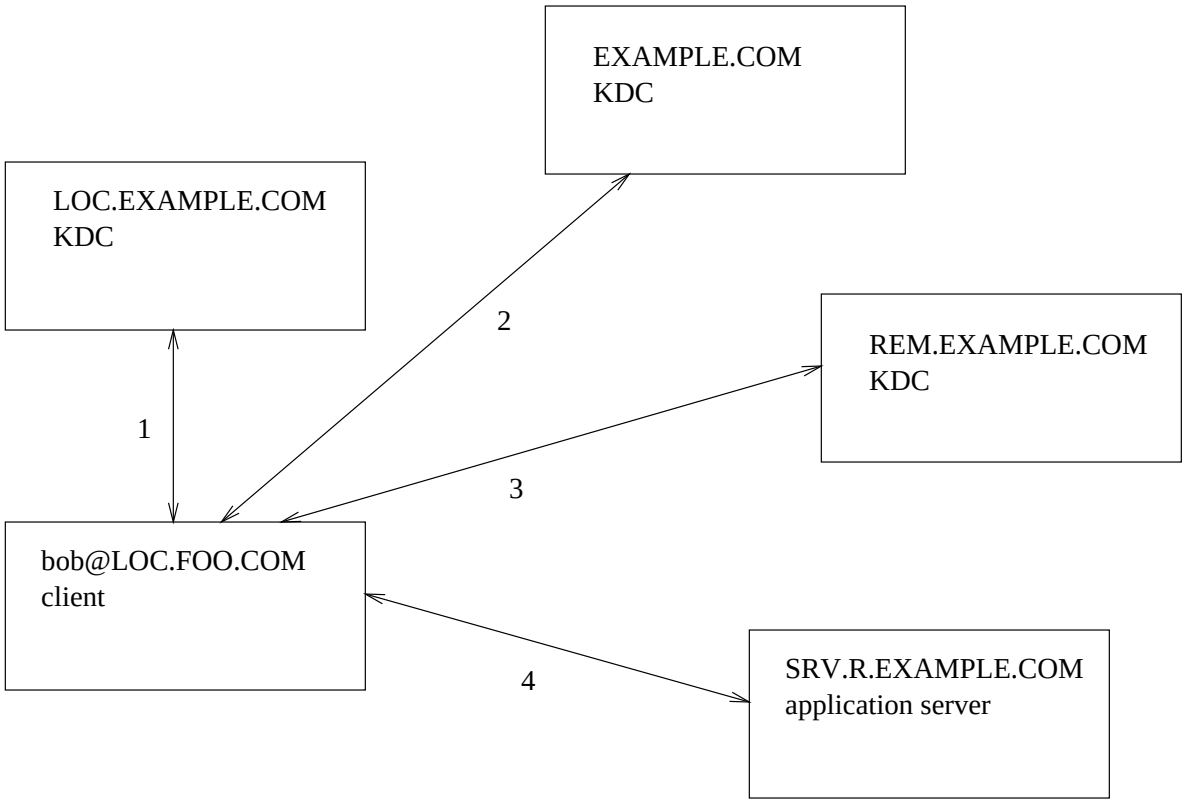
This information could be obtained from a directory lookup, or a MIT-style configuration file. If successful, the TGS reply will contain a padata field with the realm of the service. In addition, the reply contains a crossrealm ticket targetted at the next hop (EXAMPLE.COM domain in this case). Bob's client simply sends the next TGS request to EXAMPLE.COM and requests a crossrealm ticket for the referral realm. Ultimately, Bob's client obtains this ticket (from REM.EXAMPLE.COM in the next TGS reply in this case). Bob's client then requests and obtains a service ticket for srv.r.foo.com.

In summary, the major difference between the two approaches is that standard Kerberos obtains the realm information from a configuration file and the Windows 2000 approach obtains it from the KDC. The Windows 2000 Kerberos client has less intelligence and simply follows the referrals from the Kerberos KDC, whereas the MIT client knows which realm is the next hop in the realm trust graph. A benefit that the crossrealm referral feature provides is reduced product support costs. Since the configuration is created automatically during the creation of the Kerberos realm hierarchy, there is a reduced opportunity for administrator error.

4 Referral Protocol Specification

This section presents a detailed specification of the crossrealm referral protocol used in Windows 2000. The details are derived from [1] and are extensions to the core Kerberos protocol, which is described in RFC 1510 [2]. The changes fall into three categories: extensions of flag fields, extensions to the KDC reply message, and new data types to hold referral information. The extensions to the flag field are being incorporated into RFC1510bis [3].

The Windows 2000 referral protocol requires an addition to the set of options the KDC recognizes, which are defined by the KDCOptions type. The new flag, *name-canonicalize*, is used by a client to request that the KDC attempt to look the service name up in other realms in addition to the KDC's own realm. This flag



1. TGS exchange with local KDC
2. TGS exchange with EXAMPLE.COM KDC
3. TGS exchange with REM.EXAMPLE.COM KDC
4. AP exchange with SRV.R.EXAMPLE.COM server

Figure 3: Obtaining a Crossrealm Service Ticket

is sent in the kdc-options field of the TGS request message:

```
KDCOptions ::=   BIT STRING {
    reserved(0),
    forwardable(1),
    forwarded(2),
    proxiable(3),
    proxy(4),
    allow-postdate(5),
    postdated(6),
    name-canonicalize(15),
    renewable-ok(27),
    enc-tkt-in-skey(28),
    renew(30),
    validate(31)
    }
```

If the client sets the name-canonicalize bit, the KDC will attempt to first lookup the server principal name in its own realm database. If the lookup is successful, the KDC will issue a service ticket using the same steps as in the case where the name-canonicalize bit is not set. If the lookup is not successful, the server principal name may belong to a different realm. The KDC will attempt to determine the correct realm (the KDC may look in a configuration file, or use a directory lookup, or use some other method). If the KDC determines the realm, then the realm name will be returned in the encrypted-pa-data field that is the last field in the encrypted part of the TGS reply message:

```
EncKDCRepPart ::= SEQUENCE {
  key[0]                 EncryptionKey,
  last-req[1]            LastReq,
  nonce[2]               INTEGER,
  key-expiration[3]      KerberosTime
                         OPTIONAL,
  flags[4]               TicketFlags,
  authtime[5]            KerberosTime,
  starttime[6]           KerberosTime
                         OPTIONAL,
  endtime[7]             KerberosTime,
  renew-till[8]          KerberosTime
                         OPTIONAL,
  srealm[9]              Realm,
  sname[10]              PrincipalName,
  caddr[11]              HostAddresses
                         OPTIONAL,
  encrypted-pa-data[12] SEQUENCE OF
                         PA-DATA OPTIONAL
}
```

The data itself is an ASN.1 encoded structure containing the server's realm, and optionally, the canonical server principal name. The preauthentication data type is KRB5-PADATA-SERVER-REFERRAL-INFO (20) from [1].

```
KERB-PA-SERV-REFERRAL ::= SEQUENCE {
 referred-server-realm[0]  Realm,
 referred-server-name[1]   PrincipalName,
                           OPTIONAL
}
```

The client can use the referred-server-name field to see if it already has a ticket targetted at the server in its ticket cache. If so, it will not have to follow a sequence of crossrealm referrals from the KDC.

If the name-canonicalize bit in the TGS request is not set, then the KDC will only look up the name as a principal name in the request realm. Otherwise, the KDC must also check to see if the supplied service name is an alias for the canonical service name. The server principal name in both the ticket and the KDC reply must be the canonical server principal name instead of one of the aliases. This approach frees the application server from needing to know about aliases or from having to translate them into canonical names as is done in MIT Kerberos.

The client, by setting the name-canonicalize bit, indicates its willingness to receive a ticket for a principal other than the one in its TGS request message. For example, it may receive a crossrealm ticket granting ticket (TGT) to the next hop realm on the realm trust path towards the destination realm that contains the server principal name in the client's original request. This behaviour requires a change to the MIT Kerberos client, which normally rejects a TGS reply with a ticket targetted at a principal that was not the original principal in its TGS request.

The Windows 2000 KDC performs additional steps when aliases are used to ensure interoperability between MIT Kerberos and Windows 2000, which introduced new naming formats. In particular, a client may request a ticket to an alias for a service without setting the name-canonicalize bit. In this situation the KDC reply uses the alias rather than the canonical name in order to let MIT clients, which expect the reply to include the same name as the request, to function properly. The ticket, however, includes the canonical name for the service. A request with the name-canonicalize bit set, though, always returns the canonical server name.

This referral protocol allows a client to perform crossrealm authentication without knowing anything

other than the name of the realm that it belongs to. In addition, the protocol allows a host with multiple names to let services use all those names, rather than relying on either separate accounts for each hostname or a separate name canonicalization step at the client.

5 Comparison of the Two Approaches

We compare the MIT Kerberos crossrealm approach with the Windows 2000 crossrealm referral approach in the areas of security, ease of administration, and performance.

5.1 Security

The protocol change has two impacts on security: first, the client code is simpler, and therefore easier to verify. Second, the client does not need to trust DNS to perform reverse lookups to canonicalize names, which removes the need to trust the service. Our implementation of the crossrealm referral algorithm in the MIT client resulted in 150 fewer lines of code after removing the MIT Kerberos client crossrealm algorithm. The resulting client is simpler and it would therefore be easier to analyze the security of the new client. (Conversely, a KDC that implements the referral crossrealm algorithm is likely to have additional lines of code versus a KDC that implements the MIT Kerberos crossrealm algorithm).

Since the Windows 2000 KDC is doing the name canonicalization instead of the client, DNS reverse lookups are no longer necessary on the client. Thus the client does not have to trust the DNS and the network that DNS queries and responses travel over in order to be assured of mutual authentication. The Windows 2000 Kerberos approach to name canonicalization has a security advantage over the MIT Kerberos approach. The MIT Kerberos approach could be secured using DNSSEC [9], but DNSSEC is unlikely to be widely deployed in the near future.

5.2 Ease of Administration

The Windows 2000 crossrealm referral algorithm requires no configuration data on the client except the client workstation's realm name. Therefore, updates to the realm hierarchy require no updates to client configurations. Although an automatic update mechanism could be implemented on a per platform basis, such a mechanism does not exist for many platforms currently. This mechanism would also decrease performance by significantly adding to network traffic. The Windows 2000 crossrealm algorithm has a major advantage in the case where the realm hierarchy is not static. In order for name canonicalization to work properly, the KDC database principal entries must contain the additional principal alias names.

5.3 Performance

The MIT Kerberos crossrealm algorithm has an advantage with respect to performance since it more fully utilizes the Kerberos ticket cache.

The MIT Kerberos client more effectively utilizes the ticket cache since the MIT Kerberos client canonicalizes the name so there is only one name to look for in the ticket cache when searching for cached service tickets. There are two situations where these requests are not needed for MIT Kerberos clients. First, when the client has an existing ticket to the service using a different alias, it need not contact the KDC to learn that both aliases refer to the same principal. (But it would need to contact the DNS to canonicalize the alias into the canonical name that is in the cached ticket, and if DNSSEC is being used, then some of the performance difference is negated). However, this makes it difficult for the aliases to later be separated and assigned individual accounts, such as when a single process hosts multiple services and those services are then split into separate processes. The Windows 2000 implementation of Kerberos also suffers because it only caches a single alias for a service, so that if multiple aliases are regularly used then the cache will suffer from conflict misses, as each time a different alias is used the old ticket will be evicted.

We propose that referral data be returned in the following ticket extension:

```
TE-REFERRAL-DATA ::= SEQUENCE {
     ReferralRealm  Realm,
     ReferralNames  SEQUENCE OF
                    PrincipalNames
                    OPTIONAL
}
```

The sequence of principal names can include the aliases which allows the ticket cache to be more effectively utilized when referrals are being used.

The second inefficiency of the referral protocol occurs when the client already has a ticket to the realm of a service. The MIT Kerberos client does not need to contact a KDC to learn which realm that is (but it does need to contact the DNS). Similarly, if a MIT client has tickets cached for a portion of the trusted realm path, it can use those tickets rather than asking the KDC to provide a referral. The Windows 2000 protocol allows the client to bypass the trust path if it already has a ticket to the realm of the service, but this is only significant if the path is long. This performance difference is most noticeable when a client is looking up many different services in the same realm, and that realm is directly trusted by the client's realm.

6 Client Referral Handling Pseudocode Algorithm

We give the high level algorithm that we have implemented and successfully tested in the MIT Kerberos 1.1 client. The general algorithm is to initially request a ticket to the service in the client's own realm. If a service ticket is not returned, then loop and request tickets to krbtgt/referral_realm where referral_realm is the returned referral realm. When a ticket to krbtgt/referral_realm is obtained, then request a ticket to the service from a referral_realm KDC. This algorithm works since referral_realm is the destination realm, so each intermediate KDC can avoid the overhead of translating the name again.

```
start TGS_REQ creation;
orig_sname = req.sname;
set name canon bit;
send TGS_REQ;
receive and decode TGS_REP;
grab referral_realm, if present;
next_must_be_serv_tkt = FALSE;
DONE = FALSE;

while (!DONE) {
 if ( (received message is a TGS_REP) &&
     (reply.sname is not of the form
     krbtgt/realm for some realm) ) {
      we have service ticket;
      DONE = TRUE;
 }
 else if(reply.sname == 'krbtgt/realm') {
  if (next_must_be_serv_tkt == TRUE)
      go to error;
  target_realm = realm;
  if(target_realm == referral_realm) {
     next_must_be_serv_tkt = TRUE;
     create and send new TGS_REQ to
     target_realm without bit 15 using the
     ticket we just got back from the KDC
     using the original service name in the
     request.sname;
     receive and decode TGS_REP message and
     update reply.sname;
  }
  else {
     create and send new TGS_REQ to
     target_realm without bit 15 using the
     ticket we just got back from the KDC
     using the name 'krbtgt/referral_realm'
     in the request.sname field of the
     TGS_REQ;
     receive and decode TGS_REP message and
     update reply.sname;
  }
}
 else if (received msg type == KRB_ERROR)
   go to krb-error;
 else
   go to error;

}  // end while loop
```

The preceding pseudocode has the following issue with reuse of tickets in the ticket cache. If a client application obtains a service ticket, the server principal name in the ticket will be the canonical name for the principal. If the client application is using an alias to identify the server principal, the client Kerberos code will not see that the desired service ticket is already in the ticket cache when it goes to re-establish a new context with the application server. The client Kerberos code then obtains a new service ticket even though it could use the existing service ticket in the ticket cache. For example, suppose the client is using the alias ldap/ldap.example.org while the canonical name for the server is ldap/foo.example.org.

To improve ticket cache utilization, our Kerberos client makes an initial TGS request to the local realm without the name-canonicalize bit set. If the server principal not found error is returned by the KDC, the client submits a new request with the name-canonicalize bit set. Thus a client that obtains a service ticket with an alias name will be able to reuse that ticket from the ticket cache with the same alias name; the cost of this approach is one extra exchange with the KDC when initially obtaining a service ticket for services that are located outside the local realm.

A more optimal solution is to always set the name-canonicalize bit on the initial TGS request and add the alias name into the ticket cache data that is associated with the obtained service ticket. Subsequent matches when searching the ticket cache are made against the full set of names that are associated with the service ticket. This approach is used in Windows 2000. A more general discussion of this problem is in section 5.

7 Naming Issues

Here we discuss naming issues that are applicable beyond the Kerberos authentication system. Short names are the names users sometimes input to identify servers. For example, "telnet foo" could be short for "telnet foo.example.org". The client is then left with two options: send the short name to the KDC, or perform a limited amount of name canonicalization locally. The MIT Kerberos approach to name-

canonicalization solves the problem of obtaining the long name by using DNS reverse lookups. Unfortunately, this approach depends on the security of DNS. The Windows 2000 client does not canonicalize names at all, so the short name is sent to the KDC. (In the Windows 2000 case, this design decision was driven by the desire for backwards compatibility with Netbios which has a flat namespace of hostnames, thus increasing the chances that short names are unique.) The problem with not canonicalizing a short name is that it may end up being sent in a crossrealm referral request to a KDC in another realm. This may result in interoperability problems with existing KDC's; in addition, short names do not have enough information to allow remote realms to properly determine the appropriate realm to forward the request to. If DNS is used to obtain a long name for A RR lookup purposes, then it makes sense for a client to resolve the short name into a long name using its DNS resolver. This resolution is at most vulnerable to denial of service attacks if the short name is unique among the domains in the DNS completion search list on the client host. This approach makes sense when an authentication mechanism that uses X.509 certificates [10] is used, since the certificates can contain multiple names in the altSubjectName field, but the certificates will not contain short names.

In short, we argue that the optimal approach is in between the MIT Kerberos client and the Windows 2000 client (but closer to the Windows 2000 client) where the client does canonicalize short names into long names, but all other name canonicalization is left to the KDC. This canonicalization of short names applies to any authentication protocol that takes input names from users.

We make some brief remarks on name canonicalization of short names. Some resolvers may not return the name on the DNS completion search list that corresponds to the returned A RR. The following algorithm can be used to overcome this obstacle: the client should take each name on the DNS search list and append it to the input name and terminate it with a ".". This name should then be inputted into the resolver lookup call. When the function call returns an IP address, the client has obtained the corresponding long name.

8 Conclusions

We have presented the Windows 2000 crossrealm referral mechanism and compared it to the MIT Kerberos crossrealm approach. Overall, when both the name canonicalization and referral resolution problems are considered, the Windows 2000 approach has significant advantages with respect to both security and ease of administration. The MIT Kerberos client's better ticket cache utilization when alias names are used, combined with its knowledge of the realm hierarchy can be used to reduce the number of exchanges with the KDC in some cases. The performance advantage can be offset to some extent by having the KDC return a list of alias principal names in the referral data, as described in section 5.

Further work should build upon the Windows 2000 approach to improve its use of the ticket cache, possibly using additional returned names as described above. An alternative approach is to move Kerberos client configuration data into the DNS (as in [11]) and use DNSSEC [9]. In this approach, clients obtain the realm for a server principal name from a DNS TEXT resource record. This alternative approach solves the management problem, but the main issue is dependence on DNSSEC [9] which is unlikely to be widely deployed in the near future. It does not appear that the alternative approach has any performance or security advantages over the recommended approach.

Additionally, we have presented the specification and design for integration of the crossrealm referral algorithm into the MIT Kerberos client.

9 Acknowledgements

Thanks to John Brezak for information on the Windows 2000 referral implementation. There was also a discussion of these topics on the IETF CAT Working Group mailing list; we wish to thank Paul Leach and other participants for their contributions.

References

[1] M. Swift, "Generating KDC Referrals to Locate Kerberos Realms", Internet draft (work in progress), draft-swift-win2k-krb-referrals-00.txt, October 1999.

[2] J. Kohl, C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, September 1993.

[3] C. Neuman, J. Kohl, T. Ts'o, "The Kerberos Network Authentication Service (V5)," Internet draft (work in progress), draft-ietf-cat-kerberos-revisions-06.txt, July 2000.

[4] B. C. Neuman, T. T'so, "An Authentication Service For Computer Networks," IEEE Communications Magazine, Vol. 32(9), pp. 33-38, September 1994.

[5] R. M. Needham, and M. D. Schroeder, "Using Encryption for Authentication in Large Networks

of Computers," *Communications of the ACM.* 21(12):993-999, December 1978.

[6] J. G. Steiner, B. C. Neuman, J. I. Schiller, "Kerberos: An Authentication Service for Open Network Systems," pp. 191-202 in USENIX Conference Proceedings, Dallas, Texas (February 1988).

[7] M. Hur, B. Tung, T. Ryutov, C. Neuman, G. Tsudik, A. Medvinsky, B. Sommerfeld, "Public Key Cryptography for Crossrealm Authentication in Kerberos," Internet draft (work in progress), draft-ietf-cat-kerberos-pk-cross-06.txt, October 1999.

[8] P. Mockapetris, "Domain Names - Concepts and Facilities," RFC 1034, November 1987.

[9] D. Eastlake, "Domain Name System Security Extensions," RFC 2535, March 1999.

[10] R. Housley, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," RFC 2459, January 1999.

[11] K. Hornstein, J. Altman, "Distributing Kerberos KDC and Realm Information with DNS," Internet draft (work in progress), draft-ietf-cat-krb-dns-locate-02.txt, March, 2000.

A Data Structures and Code Modules for Referral Handling in the MIT Kerberos Code Base

A new data type is introduced for the service referral data returned by the KDC:

```
typedef struct _krb5_serv_referral {
    krb5_magic magic;
    krb5_principal referred_server_name;
 /* includes realm */
} krb5_serv_referral;
```

A new flag is defined to be set when obtaining referral data:

```
#define KDC_OPT_NAME_CANONICALIZE 0x00010000
```

A new return code for krb5_get_cred_via_tkt_ext() is introduced, to indicate that the referral data along with crossrealm TGT are returned:

```
#define KRB5_GOT_REFERRAL_TGT    1
```

A new type of pre-authentication data is introduced to denote that PADATA contains a service referral:

```
#define KRB5_PADATA_SERVER_REFERRAL_INFO  20
```

A new member is added to the krb5_enc_kdc_rep_part data structure to denote the PADATA containing the service referral, which is being sent back by the KDC at the end of the encrypted part of KDC reply:

```
krb5_pa_data FAR * FAR *padata;
// preauthentication referral data from KDC
```

A.1 System Flow

In the MIT kerberos client, the function krb5_get_cred_from_kdc() is used to get a ticket from the KDC. When the client needs to obtain a service ticket, the function krb5_get_cred_from_kdc() is first called in attempt the get the service ticket from the default realm. This is the original Kerberos function for obtaining the service, except it will not "walk" the realm tree and try to obtain TGT's from different realms because the configuration file doesn't specify any additional realms.

If the service is not on the default realm, krb5_get_cred_from_kdc() will fail. In this case the extended version, krb5_get_cred_from_kdc_ext() is called to obtain a service ticket by setting the name-canonicalize bit and following crossrealm referrals as described in the preceding section.

In the crossrealm case, the ticket cache is checked to see if a crossrealm ticket targetted at the referral realm is in the cache. If so, this ticket is used in a TGS request to the realm of the destination service in order to obtain a service ticket. DNS lookups to obtain DNS SRV resource records for KDC's are used to find the address of a KDC for a specific realm. All the intermediate TGT's are cached.