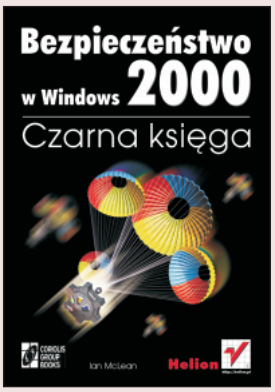
Bezpieczeństwo w Windows 2000.
Czarna księga



Autor: Ian McLean
Tłumaczenie: Andrzej Będkowski
ISBN: 83-7197-416-7
Tytuł oryginału: Windows 2000 Security. Little Black Book
Format: B5, stron: 392

Książka „Bezpieczeństwo systemu Windows 2000. Czarna księga" stanie się niezastąpionym narzędziem dla specjalistów z dziedziny zabezpieczeń zajmujących się środowiskiem systemu Windows 2000. Zawiera wiele sposobów i wskazówek dotyczących wprowadzania bezpiecznych, lecz użytecznych, zasad sieciowych. Czytelnik znajdzie w niej informacje przydatne zarówno w niewielkich sieciach (do kilkuset użytkowników), jak i w sieciach rozległych, z których korzystają wielkie firmy międzynarodowe. Autor odkrywa tajemnice usług katalogowych Active Directory, zasad grupowych, protokołów zabezpieczeń, szyfrowania, zabezpieczeń z kluczem publicznym, certyfikatów zabezpieczeń, kart elektronicznych, zabezpieczeń protokołu IP, wirtualnych sieci prywatnych i programów narzędziowych związanych z bezpieczeństwem systemu Windows 2000.

- Nie zwlekaj, lecz kup tę książkę, by:
- zastosować Active Directory do ustanawiania zasad zabezpieczeń użytkownika, grupy i komputera;
- skonfigurować elastyczne zasady zabezpieczeń za pomocą obiektów zasad grupowych;
- delegować zadania związane z bezpieczeństwem systemu bez naruszania zasad domeny;
- zainstalować systemy zabezpieczeń z kluczem publicznym i kluczem tajnym;
- zastosować protokół zabezpieczeń Kerberos 5;
- wprowadzić szyfrowanie plików i folderów oraz zasady odzyskiwania foldera zaszyfrowanego;
- zainstalować jednostkę certyfikującą i wydawać certyfikaty;
- wprowadzić zasady logowania się za pomocą kart elektronicznych;
- korzystać z narzędzi do tworzenia oprogramowania dla kart elektronicznych;
- wprowadzić zabezpieczenia internetowe za pomocą odwzorowywania certyfikatów oraz korzystania z wirtualnych sieci prywatnych;
- wprowadzić zabezpieczenia na poziomie sieci za pomoca protkołu IPSec;
- korzystać z narzędzi programowych do konfigurowania zabezpieczeń.

Oprócz wyjaśnienia powyższych zgadnień, czytelnik może liczyć na:

- proste, opisane krok po kroku, rozwiązania problemów związanych z bezpieczeństwem systemu Windows 2000;
- fachowe porady praktyczne umożliwiające osiągnięcie równowagi pomiędzy bezpieczeństwem systemu a użytecznością systemu;
- specjalistyczne wskazówki pomagające w podniesieniu kwalifikacji administratora zabezpieczeń.

Książka ta jest przeznaczona dla:

- administratorów sieci i administratorów zabezpieczeń,
- informatyków zajmujących się instalowaniem sieci bezpiecznych.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 4.

Protokoły zabezpieczeń

W tym rozdziale:

- Konfigurowanie protokołów ze wspólnymi kluczami tajnymi.
- Zastosowanie Centrum dystrybucji kluczy.
- Podstawowe wiadomości o podprotokołach protokołu Kerberos.
- Analiza biletów protokołu Kerberos.
- Delegowanie uwierzytelniania.
- Konfigurowanie zasad domeny protokołu Kerberos.
- Zastosowanie interfejsu dostawcy obsługi zabezpieczeń.

W skrócie

Protokoły

Protokoły są zasadniczą częścią każdego systemu zabezpieczeń, a podstawowe wiadomości o protokołach zastosowanych w systemie Windows 2000 są koniecznym elementem wyposażenia administratora. Jednak protokołów nie konfiguruje się od razu, jak np. *Active Directory* czy zasady grup. Przed przystąpieniem do konfigurowania protokołu, konieczna jest znajomość zasad ich działania oraz skutków, jakie pociągają za sobą wprowadzane zmiany. Niniejszy rozdział zawiera więcej teorii niż którykolwiek z pozostałych w tej książce i jednocześnie mniej gotowych rozwiązań i opisów czynności. Nie jest to bynajmniej złe — pokaźna wiedza o protokołach zabezpieczeń jest konieczna.

Zabezpieczenia warstwy transportowej oparte na protokole *Secure Sockets Layer 3* (SSL 3/TLS) opisano w rozdziale 6., przy okazji omawiania certyfikatów z kluczem publicznym. Zabezpieczenia protokołu *IP* na poziomie sieci omówiono w rozdziale 10. Pozostaje *NT LAN Manager* (NTLM) i protokół *Kerberos 5*. Protokołu *NTLM* używa się, aby zapewnić zgodność wsteczną z poprzednimi wersjami sieciowych systemów operacyjnych Microsoftu, takimi jak Windows NT 4 i LAN Manager. Tak więc większa część niniejszego rozdziału poświęcona jest protokołowi *Kerberos*.

Rozwiązania pokrewne:	Na stronie:
Ustawianie zabezpieczeń sieci Web	190

Porównanie protokołów NTLM i Kerberos

Dostawca zabezpieczeń *NTLM* w celu uwierzytelniania korzysta z procedury wyzwanie-odpowiedź. Dostawca zabezpieczeń zna hasło danego użytkownika lub, ściślej mówiąc, wyciąg uzyskany za pomocą funkcji mieszającej *MD4*. Za pomocą funkcji mieszającej *MD4* szyfruje losowo wygenerowany blok danych i odsyła go do klienta (wyzwanie — challenge). Następnie klient rozszyfrowuje blok danych i przesyła z powrotem do serwera. Jeśli klient również zna prawidłowe hasło, dane zostaną rozszyfrowane poprawnie i serwer zarejestruje danego klienta jako uwierzytelnionego. Z kolei dostawca zabezpieczeń generuje niepowtarzalny znacznik dostępu (access token), który jest przesyłany do klienta, aby korzystał z niego w przyszłości. Kolejne uwierzytelnienia klienta dokonywane są na podstawie tego znacznika i dostawca zabezpieczeń *NTLM* nie musi powtarzać procedury uwierzytelniania wyzwanie-odpowiedź.

MD2, *MD4* i *MD5* są algorytmami wyznaczania funkcji mieszającej wiadomości opracowanych dla potrzeb aplikacji tworzących podpisy cyfrowe, w których duży blok wiadomości musi zostać „skompresowany" w sposób bezpieczny, zanim zostanie podpisany za pomocą klucza prywatnego. Opis i kody źródłowe wymienionych powyżej trzech algorytmów można znaleźć w dokumentach RFC od 1319 do 1321. Więcej informacji dostępnych jest na stronie internetowej RSA Laboratory *www.rsasecurity.com/rsalabs/*, a szczególnie pod adresem *www.rsasecurity.com/rsalabs/faq/3-6-6.html*.

Protokół *Kerberos* korzysta z idei *biletów pośrednich* (proxy tickets). Weźmy pod uwagę, np. proces A, który wywołuje aplikację B. Następnie aplikacja B staje się personifikacją procesu A, tzn. aplikacja B w ustalony sposób działa jak A. Jednak jeśli B, działając jako A, wywoła inną aplikację (C), to domyślnie aplikacja C będzie personifikacją B, a nie A, ponieważ uprawnienia dotyczące zabezpieczeń procesu A nie zostaną delegowane do aplikacji C. Prawdziwe delegowanie — takie, jakie zapewnia protokół *Kerberos* — oznacza, że jeśli aplikacja B, która jest działającym wątkiem (thread) A, wywoła inną aplikację — C — to aplikacja C może być personifikacją A. Aplikacja B posiada bilet pośredni (proxy ticket) aplikacji A, który umożliwia personifikację A nawet, jeśli wywołuje inne aplikacje.

Komputery pracujące pod kontrolą systemów Windows 3.11, Windows 9x lub Windows NT 4 będą korzystały z protokołu *NTLM* przy uwierzytelnianiu sieciowym w domenach Windows 2000. Komputery pracujące pod kontrolą systemu Windows 2000 będą korzystały z protokołu *NTLM* w przypadku uwierzytelniania przez serwery systemu Windows NT 4 i przy dostępie do zasobów znajdujących się w domenach systemu Windows NT 4. Ale w systemie Windows 2000 głównym protokołem jest *Kerberos*. Korzyści z zastosowania protokołu *Kerberos* do uwierzytelniania są następujące:

- *uwierzytelnianie wzajemne* — protokół *NTLM* pozwala serwerowi na weryfikowanie tożsamości klientów, ale nie umożliwia klientom sprawdzania tożsamości serwera ani nie daje serwerowi możliwości zweryfikowania tożsamości innego serwera. Protokół *Kerberos* gwarantuje, że obydwie strony połączenia sieciowego wiedzą, że druga strona połączenia jest tym, za co się podaje,
- *szybsze połączenia* — w przypadku uwierzytelniania za pomocą protokołu *NTLM* serwer aplikacji, aby uwierzytelnić każdego klienta, musi dołączyć się do kontrolera domeny. W przypadku uwierzytelniania za pomocą protokołu *Kerberos*, serwer nie musi odwoływać się do kontrolera domeny, ale zamiast tego może uwierzytelnić klienta, sprawdzając jego dane uwierzytelniające. Klienci uzyskują dane uwierzytelniające dla danego serwera tylko raz, a potem korzystają z nich ponownie w trakcie logowania sieciowego,

- *prostsze zarządzanie relacjami zaufania* — jedną z głównych korzyści uwierzytelniania wzajemnego w przypadku protokołu *Kerberos* jest to, że relacje zaufania pomiędzy domenami systemu Windows 2000 domyślnie są obustronne i przechodnie. Jeśli sieć zawiera kilka drzew, to dane uwierzytelniające — wydane przez domenę w dowolnym drzewie — są uznawane w całym drzewie,

Przechodnia relacja zaufania oznacza, że jeśli A ufa B i B ufa C, to A ufa C. Relacje zaufania ustanawiane za pomocą protokołu *Kerberos* są przechodnie, a relacje ustanawiane za pomocą protokołu *NTLM* — nie.

- *uwierzytelnianie delegowane* — zarówno protokół *NTLM*, jak i protokół *Kerberos* dostarczają informacji potrzebnych usłudze do personifikowania swojego klienta lokalnie, ale niektóre aplikacje rozproszone zostały zaprojektowane tak, że usługa zewnętrzna musi personifikować klientów, łącząc się z usługami wewnętrznymi na innym komputerze. Protokół *Kerberos* zawiera mechanizm pośredniczący, który pozwala usłudze na personifikowanie klienta, kiedy ten łączy się z innymi usługami. Protokół *NTLM* takiego mechanizmu nie zawiera,
- *współdziałanie* — implementacja protokołu *Kerberos* w wersji firmy Microsoft jest oparta na specyfikacji przedstawionej grupie roboczej *Internet Engineering Task Force* (IETF). W wyniku tego implementacja protokołu *Kerberos* w systemie Windows 2000 stanowi podstawę współdziałania z innymi sieciami (szczególnie z sieciami zgodnymi ze standardem *POSIX*), w których do uwierzytelniania stosuje się protokół *Kerberos*.

Protokół uwierzytelniania *Kerberos* umożliwia wzajemne uwierzytelnianie pomiędzy klientem a serwerem oraz pomiędzy dwoma serwerami, zanim pomiędzy nimi zostanie ustanowione połączenie sieciowe. Protokół przyjmuje, że początkowe transakcje pomiędzy klientami a serwerami mają miejsce w otwartej sieci, w której większość komputerów nie jest zabezpieczonych i „napastnicy" mogą monitorować oraz dowolnie modyfikować pakiety tak, jak w Internecie. Protokół *Kerberos* gwarantuje, że nadawca wie, kim jest odbiorca i odwrotnie oraz gwarantuje, że nikt z zewnątrz nie będzie mógł się podawać za żadną ze stron.

Rozwiązania natychmiastowe

Konfigurowanie protokołów ze wspólnym kluczem tajnym

W protokole *Kerberos* zastosowano uwierzytelnianie wymagające wspólnych kluczy tajnych. Jeśli tylko dwoje ludzi zna klucz tajny, wówczas każdy z nich może zweryfikować tożsamość drugiej strony przez uzyskanie potwierdzenia, że osoba ta zna ten sam klucz tajny. Załóżmy, że np. Bonnie często wysyła wiadomości do Clyde'a, który musi mieć pewność, że wiadomość od Bonnie jest autentyczna, zanim zacznie działać zgodnie z otrzymanymi informacjami. Bonnie i Clyde jako rozwiązanie tego problemu wybrali hasło, które nie będzie udostępniane nikomu innemu. Jeśli z wiadomości od Bonnie będzie wynikało, że nadawca zna to hasło, Clyde będzie wiedział, że nadawcą jest Bonnie.

W jaki sposób Bonnie pokaże, że zna to hasło? Mogłaby zamieścić w zakończeniu swojej wiadomości coś na wzór bloku podpisu. Jest to sposób prosty i wydajny, ale będzie bezpieczny tylko wtedy, gdy Bonnie i Clyde będą pewni, że nikt inny nie czyta ich poczty. Niestety, wiadomości są przesyłane przez sieć, z której korzysta Szeryf, posiadający analizator sieci. Tak więc Bonnie, podając jedynie hasło, nie może udowodnić, że je zna, tylko musi wykazać się jego znajomością bez przesyłania przez sieć.

W protokole *Kerberos* rozwiązano ten problem za pomocą kryptografii z kluczem tajnym. Zamiast współdzielenia hasła, partnerzy komunikacji współdzielą klucz i stosują go do weryfikowania tożsamości drugiej strony. Klucz wspólny musi być symetryczny. Innymi słowy, ten sam klucz służy do szyfrowania i odszyfrowywania. Jedna ze stron udowadnia znajomość klucza poprzez zaszyfrowanie części danych, a druga — rozszyfrowując te dane. Uwierzytelnianie za pomocą klucza tajnego rozpoczyna się, gdy jedna ze stron przedstawi wartość uwierzytelniającą w postaci części danych zaszyfrowanych za pomocą klucza tajnego. Zestaw danych do tworzenia wartości uwierzytelniającej za każdym razem musi być inny, w przeciwnym razie stara wartość uwierzytelniająca mogłaby być wykorzystana ponownie przez kogoś, komu udałoby się podsłuchać wymianę danych. Po otrzymaniu wartości uwierzytelniającej odbiorca odszyfrowuje ją. Jeśli odszyfrowanie przebiegło pomyślnie, to odbiorca wie, że osoba przedstawiająca tę wartość zna właściwy klucz. Tylko dwoje ludzi ma właściwy klucz; odbiorca jest jedną z nich, więc osoba przedstawiająca wartość uwierzytelniającą musi być tą drugą.

Przyjęto, że dwie i tylko dwie osoby posiadają klucz tajny. Jeśli byłby on znany osobie trzeciej — przestałby być tajnym.

Jeśli dane są pobierane z zaufanego źródła, odbiorca musi upewnić się, czy źródło jest tym, czym rzekomo jest oraz czy informacje nie zostały po drodze sfałszowane. Nie jest to mało znaczący problem, ale parametry są ustalone i można znaleźć właściwe rozwiązanie. Należy wrócić uwagę, że w kontekście bezpieczeństwa nie ma rozwiązań doskonałych. W przypadku komunikacji dwukierunkowej, kiedy wymagane jest uwierzytelnianie wzajemne, pojawiają się nowe problemy, zwłaszcza gdy dochodzi do współdzielenia kluczy tajnych i konieczne jest zastosowanie nowych narzędzi do rozwiązania tych problemów.

Odbiorca, aby zapewnić uwierzytelnianie wzajemne, może wziąć część danych z pierwotnej wartości uwierzytelniającej, zaszyfrować je, tworząc nową wartość uwierzytelniającą, którą wysyła do nadawcy, który z kolei może odszyfrować wartość uwierzytelniającą odbiorcy i porównać z oryginałem. Jeśli zgadzają się, nadawca będzie wiedział, że odbiorca był w stanie odszyfrować oryginał, a więc ma właściwy klucz.

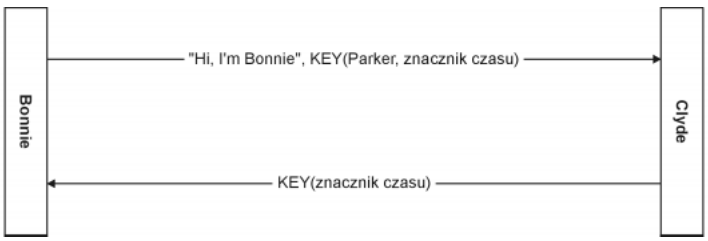
Załóżmy, np. że Bonnie i Clyde zadecydowali, iż do weryfikowania tożsamości drugiej strony, przed przesłaniem jakichkolwiek informacji, będą korzystać ze wspólnego klucza tajnego. Uzgodnili w ten sposób następujący protokół:

1. Bonnie wysyła do Clyde'a komunikat zawierający jej imię zapisane otwartym tekstem oraz wartość uwierzytelniającą zaszyfrowaną za pomocą klucza tajnego współdzielonego z Clydem. W tym protokole wartość uwierzytelniająca jest strukturą danych zawierającą dwa pola. Jedno pole zawiera informacje o Bonnie — np. nazwisko (Parker). Drugie pole zawiera aktualny czas wskazywany na stacji roboczej Bonnie.

2. Clyde otrzymuje wiadomość i za pomocą klucza współdzielonego z Bonnie odszyfrowuje wartość uwierzytelniającą i odczytuje pole zawierające czas ze stacji roboczej Bonnie.
3. Załóżmy, że oboje używają usługi sieciowej, która synchronizuje zegary w ich komputerach. Tak więc Clyde może porównać czas odczytany z wartości uwierzytelniającej z czasem wskazywanym przez zegar w jego komputerze. Jeśli różnica przekracza ustaloną granicę, Clyde odrzuca wartość uwierzytelniającą.
4. Jeśli czas ten mieści się w dopuszczalnym zakresie, to wartość uwierzytelniająca prawdopodobnie pochodzi od Bonnie, ale Clyde nadal nie ma na to dowodu. Szeryf mógł obserwować ruch w sieci i teraz powtórzył wcześniejszą próbę ustanowienia połączenia przez Bonnie. Jednak jeśli Clyde zapisał czasy uzyskane z wartości uwierzytelniających przesłanych przez Bonnie, wtedy może udaremnić próby powtarzania wcześniejszych wiadomości, odrzucając każdą wiadomość, której czas jest taki sam lub wcześniejszy niż czas ostatniej wartości uwierzytelniającej.

W poprzednim akapicie opisano atak metodą powtórzenia (replay attack). Jest to określenie warte zapamiętania, ponieważ osoby zajmujące się zabezpieczeniami poświęcają takim atakom wiele uwagi.

5. Clyde używa klucza współdzielonego z Bonnie do zaszyfrowania czasu uzyskanego z komunikatu przesłanego przez Bonnie i wysyła go do niej z powrotem. Należy zwrócić uwagę, że Clyde nie odsyła wszystkich informacji zawartych w wartości uwierzytelniającej, ale tylko czas. Gdyby odesłał wszystko, Bonnie nie miałaby sposobu, aby dowiedzieć się, czy ktoś udający Clyde'a po prostu nie skopiował wartości uwierzytelniającej z oryginalnego komunikatu Bonnie i nie wysłał do niej w niezmienionej postaci. Clyde wybrał czas, ponieważ jest to jedyna niepowtarzalna w komunikacie część informacji, którą przesłała Bonnie.
6. Bonnie otrzymuje odpowiedź od Clyde'a, odszyfrowuje ją i porównuje wynik z czasem zamieszczonym w wartości uwierzytelniającej , którą wysłała. Jeśli czas zgadza się, Bonnie może być pewna, że jej wartość uwierzytelniająca dotarła do kogoś, kto zna klucz tajny, konieczny do odszyfrowania tej wartości i oczytania czasu. Klucz tajny Bonnie współdzieli tylko z Clyde'm., więc to on odebrał jej komunikat i odpowiedział. Zarówno nadawca, jak i odbiorca mogą mieć zaufanie co do danego połączenia. Na rysunku 4.1 przedstawiono proces wzajemnego uwierzytelniania.



Rysunek 4.1. *Uwierzytelnianie wzajemne*

Zastosowanie Centrum dystrybucji kluczy

Poprzednia procedura ma pewien mankament — nie wyjaśnia jak ani skąd Bonnie i Clyde uzyskali klucz tajny, z którego korzystają podczas sesji. Dla jasności na potrzeby poprzedniej procedury przyjęto, że Bonnie i Clyde są ludźmi, którzy mogliby się spotkać i uzgodnić współdzielony klucz tajny. W rzeczywistości Bonnie może być programem-klientem pracującym na stacji roboczej, a Clyde — usługą uruchomioną na serwerze sieciowym. Następnym problemem jest to, że klient, Bonnie, komunikuje się z wieloma serwerami, więc potrzebne są klucze dla każdego z nich, a Clyde komunikuje się z wieloma klientami i również konieczne są klucze dla każdej sesji. Jeśli klient ma mieć klucz do każdej usługi, a każda usługa ma mieć klucz dla każdego klienta, to dystrybucja kluczy może szybko skomplikować się i stanowić znaczące zagrożenie.

W niniejszym rozdziale i w innych fragmentach książki, w których używa się przykładu Bonnie i Clyde'a, określa się ich jako „ona" i „on". Jak podano w poprzednim fragmencie Bonnie i Clyde mogą być programami lub usługami, ale analogia nadal jest ważna. Używanie określeń „ona" i „on" nie musi oznaczać, że Bonnie i Clyde są ludźmi.

Protokół *Kerberos* rozwiązuje ten problem, określając trzy jednostki: klienta, serwer i zaufanym urzędem niezależnym, która jest pośrednikiem pomiędzy nimi. Ten zaufany pośrednik nosi nazwę *Centrum dystrybucji kluczy* (Key Distribution Center — KDC).

Obecnie zwiększa się zapotrzebowanie na niezależne firmy dostarczające klucze tajne i zarządzające nimi, które działają jako zaufani pośrednicy. Jeśli jesteś przedsiębiorczy i bierzesz pod uwagę możliwość działalności komercyjnej, warto to przemyśleć.

Usługa *Centrum dystrybucji kluczy* jest uruchomiona na serwerze, który jest fizycznie zabezpieczony i obsługuje bazę danych o kontach wszystkich obiektów typu *security principal* w swoim obszarze, który jest w protokole *Kerberos* odpowiednikiem domeny systemu Windows 2000. Razem z informacjami o każdym z obiektów typu *security principal*, *Centrum dystrybucji kluczy* przechowuje również klucz szyfru (cryptographic key), znany tylko obiektowi typu *security principal* i *Centrum dystrybucji kluczy*. Klucz ten jest używany przy wymianie informacji pomiędzy obiektem zabezpieczeń głównych (*security principal*) i centrum oraz jest określany jako *klucz długoterminowy* (long term key). Zazwyczaj uzyskuje się go na podstawie hasła logowania obiektu typu *security principal*.

Zamknij serwer *Centrum dystrybucji kluczy* w zabezpieczonym pokoju i nie dołączaj klawiatury ani monitora. Administruj tym serwerem ze stacji roboczej-klienta z zainstalowanym odpowiednim oprogramowaniem. Dokonuj inspekcji każdego dostępu lub próby dostępu do serwera. Jeśli twoje *Centrum dystrybucji kluczy* nie jest bezpieczne, to nic innego nie jest bezpieczne.

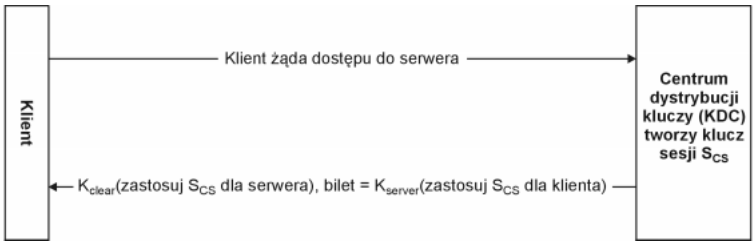
Klient, który chce skomunikować się z serwerem, wysyła żądanie do *Centrum dystrybucji kluczy*, które przydziela obu stronom unikatowy, krótkoterminowy (short term) klucz sesji, służący wzajemnemu uwierzytelnianiu. Kopia klucza sesji serwera jest zaszyfrowana w kluczu długoterminowym serwera. Kopia klucza sesji klienta jest zaszyfrowana w kluczu długoterminowym klienta.

Teoretycznie *Centrum dystrybucji kluczy* mogłoby wysyłać klucz sesji bezpośrednio do każdego obiektu typu *security principal*, którego to dotyczy, ale w praktyce taka procedura byłaby skrajnie niewydajna. Wymagałaby od serwera zachowywania w pamięci swojego klucza sesji podczas oczekiwania na wywołanie przez klienta. Ponadto serwer musiałby pamiętać klucze dla wszystkich klientów, którzy mogliby żądać obsługi. Zarządzanie kluczami zajęłoby niewspółmiernie dużą część zasobów serwera i nie dałoby możliwości rozbudowy. Protokół *Kerberos* rozwiązuje ten problem za pomocą biletów sesji (session tickets).

Zastosowanie biletów sesji

Centrum dystrybucji kluczy odpowiada na żądanie klienta, wysyłając do niego kopię klucza sesji serwera i klucza sesji klienta, jak przedstawiono na rysunku 4.2. Kopia klucza sesji klienta jest zaszyfrowana za pomocą klucza, który centrum dystrybucji współdzieli z klientem. Kopia klucza sesji serwera jest umieszczona razem z informacjami o kliencie w strukturze danych o nazwie bilet sesji (session ticket), a cała struktura jest zaszyfrowana za pomocą klucza, który *Centrum dystrybucji kluczy* współdzieli z serwerem. Klient korzysta z biletu sesji, kiedy kontaktuje się z serwerem.

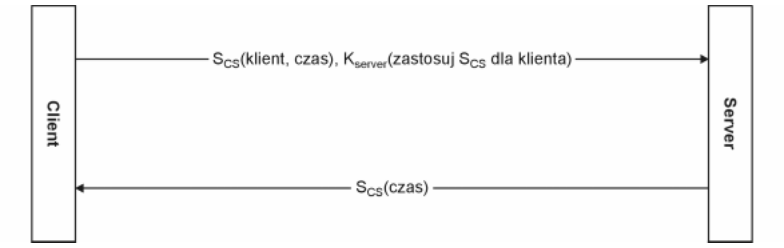
Rysunek 4.2.
Dystrybucja kluczy



Centrum dystrybucji kluczy nie śledzi swoich komunikatów, aby w ten sposób upewnić się, czy dotarły pod właściwy adres — po prostu wydaje bilet. Jeśli komunikat wysłany przez centrum KDC dostanie się w niepowołane ręce, bezpieczeństwo nie jest zagrożone. Tylko ktoś, kto zna klucz tajny klienta może odszyfrować kopię klienta klucza sesji i tylko ktoś, kto zna klucz tajny serwera, może odczytać zawartość biletu.

Klient wyodrębnia bilet i kopię klienta klucza sesji i umieszcza je w zabezpieczonej pamięci podręcznej, znajdującej się w pamięci operacyjnej, a nie na dysku. Kiedy klient chce połączyć się serwerem, wysyła komunikat zawierający bilet z wartością uwierzytelniającą zaszyfrowane za pomocą klucza sesji. Bilet, nadal zaszyfrowany za pomocą klucza tajnego serwera i wartość uwierzytelniająca razem tworzą dane uwierzytelniające klienta dla serwera.

Serwer odszyfrowuje bilet sesji za pomocą klucza tajnego, uzyskując w ten sposób klucz sesji i stosuje go do odszyfrowania wartości uwierzytelniającej klienta. Jeśli wszystko się zgadza, serwer wie, że dane uwierzytelniające klienta zostały wydane przez centrum KDC, które jest jednostką godną zaufania. Jeśli konieczne jest uwierzytelnianie wzajemne, serwer używa swojej kopii klucza sesji do zaszyfrowania znacznika czasu (timestamp) zawartego w wartości uwierzytelniającej klienta, a wynik przesyła z powrotem do klienta jako wartość uwierzytelniającą serwera. Czyni to w sposób zaprezentowany na rysunku 4.3.



Rysunek 4.3. *Uwierzytelnianie wzajemne (klient-serwer)*

Należy zwrócić uwagę, że serwer nie musi przechowywać klucza sesji, którego używa do komunikacji z klientem. Klient jest odpowiedzialny za przechowywanie biletu dla serwera w pamięci podręcznej danych uwierzytelniających i przedstawianie biletu za każdym razem, gdy chce uzyskać dostęp do serwera. Kiedy serwer nie potrzebuje już dłużej klucza sesji, może go odrzucić.

Również klient nie musi zwracać się do *Centrum dystrybucji kluczy* za każdym razem, gdy chce uzyskać dostęp do konkretnego serwera, ponieważ bilety sesji mogą być wykorzystane ponownie, o ile nie utraciły ważności. Okres ważności biletu zależy od zasad protokołu *Kerberos* (Kerberos policy) obowiązujących w danej domenie. Zwykle bilety są ważne 8 godzin. Kiedy użytkownik wylogowuje się z komputera-klienta, pamięć podręczna danych uwierzytelniających jest opróżniana, a wszystkie bilety sesji i klucze sesji są niszczone.

Zastosowanie biletu przyznającego bilet

Dla przykładu przyjmijmy, że Bonnie się loguje. Najpierw klient *Kerberos*, działający na stacji roboczej, zamienia swoje hasło na klucz kryptograficzny (cryptographic key), stosując jednokierunkową funkcję mieszającą Protokół *Kerberos* korzysta z algorytmu funkcji mieszającej *DES-CBC-MD5*, chociaż dopuszcza się stosowanie innych algorytmów. Następnie klient *Kerberos* żąda biletu sesji i klucza sesji, które może zastosować w kolejnych transakcjach pomiędzy klientem a *Centrum dystrybucji kluczy* podczas danej sesji logowania. Centrum KDC szuka w swojej bazie danych Bonnie i odczytuje z odpowiedniego rekordu bazy danych klucza długoterminowego Bonnie. Proces ten, czyli wyznaczanie kopii klucza na podstawie hasła i pobieranie kolejnej kopii klucza z bazy danych, ma miejsce tylko raz — gdy użytkownik loguje się do sieci.

Centrum dystrybucji kluczy odpowiada na żądanie klienta, zwracając specjalny rodzaj biletu sesji, który nosi nazwę biletu przyznającego bilet (Ticket Granting Ticket — TGT). Tak, jak zwykły bilet sesji, bilet przyznający bilet zawiera kopię klucza sesji, który dana usługa, w tym przypadku *Centrum dystrybucji kluczy* będzie stosować do komunikacji z klientem. Pakiet, który zwraca klientowi bilet przyznający bilet, zawiera także kopię klucza sesji, który klient może stosować do komunikacji z centrum (KDC). Bilet przyznający bilet jest zaszyfrowany w kluczu długoterminowym centrum (KDC), a kopia klienta klucza sesji jest zaszyfrowana w kluczu długoterminowym użytkownika.

W kolejnych wymianach komunikatów z centrum klient korzysta z klucza sesji. Klucz kryptograficzny uzyskany na podstawie hasła użytkownika nie jest potrzebny i może być odrzucony. Tak, jak inne klucze sesji, klucz ten jest tymczasowy, ważny tylko do czasu utraty ważności przez bilet przyznający bilet lub wylogowania się przez danego użytkownika. Z tego względu nazywany jest kluczem sesji logowania (logon session key).

Klient, przed podjęciem próby połączenia z usługą, szuka w buforze danych uwierzytelniających biletu sesji do danej usługi. Jeśli takiego biletu nie ma, klient szuka w buforze danych uwierzytelniających bilet przyznający bilet. Jeśli ten klucz zostanie znaleziony, klient pobiera z bufora właściwy klucz sesji logowania, używa go do przygotowania wartości uwierzytelniającej i wysyła wartość uwierzytelniającą oraz bilet przyznający bilet do *Centrum dystrybucji kluczy* razem z żądaniem biletu sesji dla tej usługi. Tak więc uzyskanie dostępu do centrum nie różni się niczym od uzyskiwania dostępu do pozostałych usług w domenie — wymagany jest klucz sesji, wartość uwierzytelniająca i bilet, w tym przypadku jest to bilet przyznający bilet.

Podstawowe wiadomości o podprotokołach protokołu Kerberos

Protokół *Kerberos* składa się z trzech podprotokołów:

- *Authentication Service (AS) Exchange* — stosowany przez *Centrum dystrybucji kluczy* do przydzielania klientowi klucza sesji logowania i biletu przyznającego bilet,
- *Ticket Granting Service (TGS) Exchange* — stosowany przez centrum do przydzielania usłudze klucza sesji usługi (service session key) i biletu sesji,
- *Client-Server (CS) Exchange* — stosowany, gdy klient przedstawia bilet sesji, aby uzyskać dostęp do usługi.

Aby zrozumieć, w jaki sposób te trzy podprotokoły pracują razem, zobaczmy w jaki sposób Bonnie — użytkownik stacji roboczej uzyskuje dostęp do Clyde'a — usługi sieciowej.

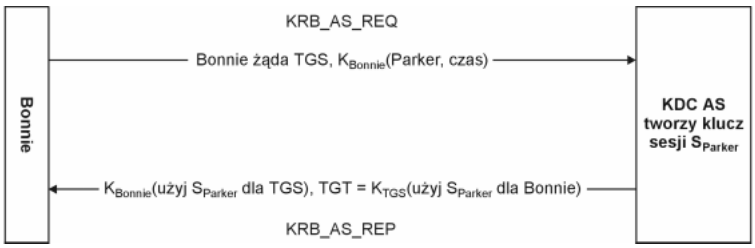
Podprotokół Authentication Service Exchange

Wymiana informacji za pomocą podprotokołu *Authentication Service Exchange* przebiega następująco:

1. Bonnie loguje się w sieci. Klient *Kerberos* na stacji roboczej Bonnie zamienia jej hasło na klucz szyfru i zapisuje go w buforze danych uwierzytelniających.
2. Klient *Kerberos* wysyła do *Centrum dystrybucji kluczy* komunikat *Kerberos Authentication Service Request* (*KRB_AS_REQ*). Pierwsza część tego komunikatu identyfikuje użytkownika, Bonnie i nazwę usługi, dla której żąda się danych uwierzytelniających — usługi wydającej bilety. Druga część komunikatu zawiera dane do uwierzytelniania wstępnego, które potwierdzają, że Bonnie zna hasło. Jest to zwykle znacznik czasu zaszyfrowany za pomocą klucza długoterminowego Bonnie.
3. *Centrum dystrybucji kluczy* odbiera komunikat *KRB_AS_REQ*.
4. *Centrum dystrybucji kluczy* wyszukuje w swojej bazie danych użytkownika o nazwie Bonnie, odczytuje jej klucz długoterminowy, odszyfrowuje dane do uwierzytelniania wstępnego i ocenia zawarty w nich znacznik czasu. Jeśli znacznik czasu jest do przyjęcia, *Centrum dystrybucji kluczy* ma pewność, że dane do uwierzytelniania wstępnego zostały zaszyfrowane za pomocą klucza długoterminowego Bonnie i że klient jest autentyczny.

5. Centrum tworzy dane uwierzytelniające, które klient *Kerberos* na stacji roboczej Bonnie może przedstawić usłudze wydawania biletów (Ticket Granting Sernice — TGS):
   - tworzy klucz sesji logowania i szyfruje jego kopię za pomocą klucza długoterminowego Bonnie,
   - umieszcza w bilecie przyznającym bilet jeszcze jedną kopię klucza sesji logowania razem z innymi informacjami dotyczącymi Bonnie, takimi jak jej dane uwierzytelniające,
   - szyfruje bilet przyznający bilet za pomocą własnego klucza długoterminowego,
   - wysyła zaszyfrowany klucz sesji logowania i bilet przyznający bilet z powrotem do klienta w komunikacie *Kerberos Authentication Service Reply* (*KRB_AS_REP*).
6. Klient po odebraniu komunikatu korzysta z klucza, uzyskanego na podstawie hasła Bonnie, do odszyfrowania jej klucza sesji logowania i zapisuje go w buforze danych uwierzytelniających. Następnie wyodrębnia z komunikatu bilet przyznający bilet i również zapisuje w tym buforze.

Działanie podprotokołu *Authentication Service Exchange* ilustruje rysunek 4.4.



Rysunek 4.4. *Wymiana komunikatów podprotokołu Authentication Service (AS)*

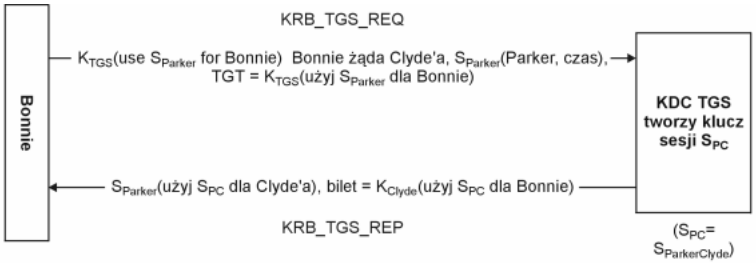
Podprotokół Ticket Granting Service Exchange

Po zakończeniu wymiany informacji przez usługę uwierzytelniania, proces wymiany komunikatów usługi wydawania biletów przebiega w następujący sposób:

1. Klient *Kerberos* na stacji roboczej Bonnie żąda danych uwierzytelniających dla usługi Clyde, wysyłając do *Centrum dystrybucji kluczy* komunikat *Kerberos Ticket Granting Service Request* (*KRB_TGS_REQ*). Komunikat ten zawiera nazwę użytkownika, wartość uwierzytelniającą zaszyfrowaną za pomocą klucza sesji logowania użytkownika, bilet przyznający bilet, uzyskany w procesie wymiany komunikatów usługi uwierzytelniania oraz nazwę usługi, dla której użytkownikowi potrzebny jest bilet.
2. Centrum (KDC) odbiera komunikat *KRB_TGS_REQ* i odszyfrowuje bilet przyznający bilet za pomocą własnego klucza tajnego, odczytując klucz sesji logowania Bonnie, który jest używany do odszyfrowania wartości uwierzytelniającej.
3. Jeśli wartość uwierzytelniająca jest poprawna, to *Centrum dystrybucji kluczy* odczytuje z biletu przyznającego bilet dane autoryzacji Bonnie i tworzy dla klienta (Bonnie) klucz sesji, który jest współdzielony z usługą (Clyde).

4. Centrum (KDC) szyfruje jedną kopię tego klucza sesji za pomocą klucza sesji logowania Bonnie, umieszcza inną kopię w bilecie razem z danymi autoryzacji Bonnie i szyfruje ten bilet za pomocą klucza długoterminowego Clyde'a.
5. KDC wysyła te dane uwierzytelniające z powrotem do klienta w komunikacie *Kerberos Ticket Granting Service Reply* (*KRB_TGS_REP*).
6. Kiedy klient otrzyma odpowiedź, korzysta z klucza sesji logowania Bonnie do odszyfrowania klucza sesji używanego w przypadku danej usługi i zapisuje ten klucz w buforze danych uwierzytelniających. Następnie wyodrębnia bilet do danej usługi i także zapisuje to w buforze.

Działanie podprotokołu Ticket Granting Service Exchange ilustruje rysunek 4.5.



Rysunek 4.5. *Komunikaty podprotokołu Ticket Granting Service Exchange*

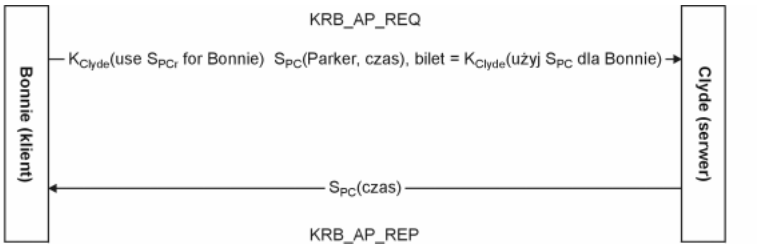
Podprotokół Client-Server Exchange

Po zakończeniu wymiany komunikatów usługi wydawania biletów rozpoczyna się procedura wymiany informacji klient-serwer (CS information exchange):

1. Klient *Kerberos* na stacji roboczej Bonnie żąda obsługi od Clyde'a, wysyłając mu komunikat *Kerberos Application Requests* (*KRB_AP_REQ*). Komunikat ten zawiera wartość uwierzytelniającą zaszyfrowaną za pomocą klucza sesji danej usługi, bilet, uzyskany podczas wymiany informacji TGS oraz wstępnie skonfigurowany znacznik wskazujący, czy klient wymaga uwierzytelniania wzajemnego.
2. Clyde odbiera komunikat *KRB_AP_REQ*, odszyfrowuje bilet i odczytuje dane autoryzacji Bonnie oraz klucz sesji.
3. Clyde korzysta z klucza sesji do odszyfrowania wartości uwierzytelniającej Bonnie i ocenia zawarty w niej znacznik czasu.
4. Jeśli wartość uwierzytelniająca jest prawidłowa, Clyde szuka w komunikacie klienta znacznika uwierzytelniania wzajemnego.
5. Jeśli znacznik jest ustawiony, usługa korzysta z klucza sesji do zaszyfrowania czasu zawartego w wartości uwierzytelniającej Bonnie, a wynik zwraca w komunikacie *Kerberos Application Reply* (*KRB_AP_REP*).
6. Gdy klient *Kerberos* na stacji roboczej Bonnie odbiera komunikat *KRB_AP_REP*, odszyfrowuje wartość uwierzytelniającą Clyde'a za pomocą klucza sesji współdzielonego z Clydem i porównuje czas przesłany przez usługę z czasem zawartym w pierwotnej wartości uwierzytelniającej klienta.

7. Jeśli czas jest ten sam, klient wie, że jest to usługa prawidłowa i połączenie jest utrzymane. W trakcie połączenia klucz sesji może być użyty do szyfrowania danych aplikacji lub też klient i serwer mogą w tym celu współdzielić inny klucz.

Na rysunku 4.6 przedstawiono proces wymiany komunikatów CS.



Rysunek 4.6. *Wymiana komunikatów CS*

Uwierzytelnianie logowania

Podczas logowania do sieci, w której do uwierzytelniania stosuje się protokół *Kerberos*, najpierw otrzymuje się bilet przyznający bilet, który przedstawia się, żądając biletów sesji dla innych usług w domenie. Podczas logowania do domeny Windows 2000 zawsze konieczny jest przynajmniej jeden bilet sesji — bilet do komputera, do którego użytkownik się loguje. Komputery pracujące pod kontrolą systemu Windows 2000 mają własne konta w domenie, które należy traktować jako konta usług. Użytkownicy interaktywni mogą mieć dostęp do zasobów w domenie Windows 2000 przez wysyłanie żądania do usługi Workstation danego komputera, ale użytkownicy zdalni wysyłają żądania do usługi *Server* tego komputera. Przed uzyskaniem dostępu do którejkolwiek z wyżej wymienionych usług lub jakiejkolwiek innej usługi, pracującej jako system lokalny, należy przedstawić bilet sesji dla danego komputera.

Przypuśćmy, że Bonnie ma konto sieciowe w domenie o nazwie *Zachód* i loguje się do komputera o nazwie Clyde, który również ma konto w domenie *Zachód*. Bonnie rozpoczyna od sekwencji *Secure Attention Sequence* (SAS) *Ctrl+Alt+Delete*. Usługa *WinLogon* na komputerze o nazwie Clyde, przełącza się na pulpit logowania i uzyskuje dostęp do biblioteki dołączanej dynamicznie (DLL) *Graphical Identification and Authentication* (GINA), *msgina.dll*, która odpowiada za zbieranie od użytkowników danych logowania, pakowanie ich do struktury danych i wysyłanie całości do *Lokalnego źródła zabezpieczeń* (Local Security Authority — LSA) w celu weryfikacji.

Bonnie wpisuje swoją nazwę użytkownika oraz hasło i z listy rozwijalnej wybiera domenę *Zachód*. Po naciśnięciu *OK*. biblioteka *DLL MSGINA* zwraca informacje logowania Bonnie do usługi *WinLogon*, która — korzystając z wywołania *API LsaLogonUser* — wysyła dane do *Lokalnego źródła zabezpieczeń* (LSA) w celu sprawdzenia.

Lokalne źródło zabezpieczeń (LSA), po otrzymaniu struktury danych z danymi logowania Bonnie, natychmiast zamienia niezabezpieczone, podane zwykłym tekstem, hasło na klucz tajny za pomocą jednokierunkowej funkcji mieszającej. Wynik zostanie zapisany w buforze danych uwierzytelniających, skąd może być pobrany w razie potrzeby do odnowienia biletu przyznającego bilet lub uwierzytelniania za pomocą protokołu *NTLM* w przypadku serwerów, dla których nie można dokonać autoryzacji za pomocą protokołu *Kerberos*.

Aby sprawdzić informacje logowania Bonnie i ustanowić dla niej sesję logowania na danym komputerze, *Lokalne źródło zabezpieczeń* musi otrzymać bilet przyznający bilet, który jest odpowiedni do uzyskania dostępu do usługi przyznającej bilet oraz bilet sesji odpowiedni do uzyskania dostępu do komputera. *Lokalne źródło zabezpieczeń* otrzymuje te bilety, korzystając z usługi *Dostawca obsługi zabezpieczeń* (Security Support Provider — SSP), która wymienia komunikaty bezpośrednio z *Centrum dystrybucji kluczy* w domenie *Zachód*.

Kolejność komunikatów jest następująca:

1. *Lokalne źródło zabezpieczeń* (LSA) wysyła komunikat *KRB_AS_REQ* do *Centrum dystrybucji kluczy* w domenie *Zachód*. Komunikat zawiera:
   - nazwę główną użytkownika, Bonnie,
   - nazwę domeny, *Zachód*,
   - dane uwierzytelnienia wstępnego zaszyfrowane za pomocą klucza tajnego uzyskanego z hasła Bonnie.
2. Jeśli dane uwierzytelniania wstępnego są poprawne, wówczas centrum (KDC) odpowiada komunikatem *KRB-AS_REP*. Komunikat ten zawiera:
   - klucz sesji dla Bonnie, który będzie współdzielony z *Centrum dystrybucji kluczy*, zaszyfrowany za pomocą klucza tajnego, uzyskanego z hasła Bonnie,
   - bilet przyznający bilet dla *Centrum dystrybucji kluczy* w domenie *Zachód* zaszyfrowany za pomocą klucza tajnego *Centrum dystrybucji kluczy*. Bilet przyznający bilet zawiera klucz sesji dla centrum (KDC), który będzie współdzielony z Bonnie oraz dane autoryzacji dla Bonnie. Dane te zawierają identyfikator zabezpieczeń (Security Identifier — SID) dla konta Bonnie, identyfikatory zabezpieczeń dla grup zabezpieczeń w domenie *Zachód*, do której należy Bonnie oraz identyfikatory zabezpieczeń dla grup uniwersalnych (universal groups) w przedsiębiorstwie, do których należy Bonnie lub jedna z jej grup domeny.
3. *Lokalne źródło zabezpieczeń* (LSA) wysyła do *Centrum dystrybucji kluczy* w domenie *Zachód* komunikat *KRB_TGS_REQ*. Komunikat ten zawiera:
   - nazwę komputera docelowego, Clyde,
   - nazwę domeny komputera docelowego, *Zachód*,
   - bilet przyznający bilet Bonnie,
   - wartość uwierzytelniającą zaszyfrowaną za pomocą klucza sesji, który Bonnie współdzieli z centrum (KDC).
4. *Centrum dystrybucji kluczy* odpowiada komunikatem *KRB_TGS_REP*. Komunikat ten zawiera:
   - klucz sesji dla Bonnie, który będzie współdzielony z Clydem., zaszyfrowany za pomocą klucza sesji Bonnie, który użytkownik ten współdzieli z *Centrum dystrybucji kluczy*,
   - bilet sesji Bonnie do komputera o nazwie Clyde, zaszyfrowany za pomocą klucza tajnego, który Clyde współdzieli z *Centrum dystrybucji kluczy*. Bilet sesji zawiera klucz sesji dla komputera o nazwie Clyde, skopiowany z biletu przyznającego bilet Bonnie, który będzie współdzielony z Bonnie oraz dane autoryzacji.

Gdy *Lokalne źródło zabezpieczeń* (LSA) otrzymuje bilet sesji Bonnie, odszyfrowuje go za pomocą klucza tajnego komputera i odczytuje dane autoryzacji Bonnie. Następnie wysyła zapytania do lokalnej bazy danych SAM (Security Accounts Manager) aby poszukać, czy Bonnie jest członkiem lokalnej grupy zabezpieczeń na tym komputerze i czy nadano temu użytkownikowi uprawnienia specjalne na danym komputerze lokalnym. Każdy z identyfikatorów zabezpieczeń, zwrócony przez kwerendę, jest dodawany do listy uzyskanej z danych autoryzacji biletu. Cała lista jest następnie używana do budowania znacznika dostępu, a uchwyt do znacznika dostępu jest zwracane do usługi *WinLogon* razem z identyfikatorem sesji logowania Bonnie i potwierdzeniem, że informacje logowania są prawidłowe.

WinLogon dla użytkownika o nazwie Bonnie tworzy stację (window station) i kilka obiektów pulpitu (desktop objects), dołącza znacznik dostępu Bonnie i uruchamia powłokę (shell process), z której Bonnie korzysta przy pracy interaktywnej z komputerem. Każda aplikacja, którą Bonnie uruchomi w trakcie swojej sesji logowania, dziedziczy jej znacznik dostępu.

Należy zwrócić uwagę, że w trakcie procesu opisanego poprzednio ten sam klucz jest używany zarówno do szyfrowania, jak i odszyfrowania. Z tego powodu wspólne klucze tajne stosowane do logowania za pomocą hasła są symetryczne.

Logowanie za pomocą kart inteligentnych

Do obsługi logowania za pomocą kart inteligentnych w systemie Windows 2000 zaimplementowano rozszerzenie zakresu funkcji wstępnej wymiany komunikatów usługi uwierzytelniania protokołu *Kerberos*, które dotyczy stosowania klucza publicznego. Kryptografia klucza publicznego jest niesymetryczna, tzn. konieczne są dwa klucze; jeden do szyfrowania, a drugi do odszyfrowania. Razem klucze te tworzą parę kluczy prywatny-publiczny. Klucz prywatny jest znany tylko właścicielowi danej pary kluczy i nie jest nigdy współdzielony. Klucz publiczny może być znany każdemu, z kim właściciel będzie wymieniał poufne informacje.

Kiedy zamiast hasła stosuje się kartę inteligentną para kluczy prywatny-publiczny zapisana na karcie inteligentnej użytkownika zastępuje współdzielony klucz tajny uzyskany z hasła użytkownika. W przypadku rozszerzenia protokołu *Kerberos*, dotyczącego klucza publicznego, wstępna wymiana komunikatów usługi uwierzytelniania jest zmodyfikowana w ten sposób, że *Centrum dystrybucji kluczy* szyfruje klucz sesji logowania użytkownika za pomocą publicznej części pary kluczy prywatny-publiczny użytkownika. Klient odszyfrowuje klucz sesji logowania za pomocą prywatnej połowy pary kluczy prywatny-publiczny.

Logowanie za pomocą kart inteligentnych przebiega w następujący sposób:

1. Użytkownik wkłada kartę inteligentną do czytnika kart dołączonego do danego komputera. Jeśli komputery z systemem Windows 2000 są skonfigurowane tak, aby można było korzystać z logowania za pomocą kart inteligentnych, włożenie karty jest równoważne sekwencji *Secure Attention Sequence* (SAS).
2. *WinLogon* wysyła komunikat do biblioteki DLL o nazwie *MSGINA*, która wyświetla okno dialogowe Informacje logowania.

3. Użytkownik wpisuje numer *PIN* (Personal Identification Number).
4. Biblioteka *DLL MSGINA* wysyła informacje logowania użytkownika do lokalnego źródła zabezpieczeń, korzystając z wywołania *API LsaLogonUser*.
5. *Lokalne źródło zabezpieczeń* (LSA) korzysta z numeru *PIN*, aby uzyskać dostęp do karty inteligentnej, na której zapisany jest klucz prywatny użytkownika oraz certyfikat *X.509v3*, zawierający publiczną połowę z pary kluczy prywatny-publiczny. Wszystkie operacje kryptograficzne korzystające z tych kluczy mają miejsce na karcie inteligentnej.
6. *Dostawca obsługi zabezpieczeń* (SSP) *Kerberos* na komputerze klienta wysyła w komunikacie żądania uwierzytelnienia wstępnego, *KRB_AS_REQ*, certyfikat z kluczem publicznym użytkownika do *Centrum dystrybucji kluczy* jako dane uwierzytelniania wstępnego.
7. *Centrum dystrybucji kluczy* sprawdza poprawność certyfikatu i odczytuje klucz publiczny, który używa do szyfrowania klucza sesji logowania. W komunikacie wysyłanym w odpowiedzi do klienta, *KRB_AS_REP*, zwraca zaszyfrowany klucz sesji logowania i bilet przyznający bilet.
8. Jeśli klient posiada prywatną połowę z pary kluczy prywatny-publiczny, korzysta z klucza prywatnego do odszyfrowania klucza sesji logowania.

Następnie zarówno klient, jak i *Centrum dystrybucji kluczy* korzystają z klucza sesji logowania do wymiany dwustronnej informacji. Nie ma innych odchyleń od protokołu standardowego.

Dostawca obsługi zabezpieczeń Kerberos (Kerberos SSP), działający na komputerze klienta, domyślnie wysyła dane uwierzytelniania wstępnego do *Centrum dystrybucji kluczy* w postaci zaszyfrowanego znacznika czasu. W systemach, w których zastosowano logowanie za pomocą kart inteligentnych, *Dostawca obsługi zabezpieczeń Kerberos* wysyła dane uwierzytelniania wstępnego w postaci klucza publicznego. W rozdziale 6. znajduje się dokładniejszy opis kluczy publicznych, a w rozdziale 9. opisano szerzej korzystanie z kart inteligentnych.

Uwierzytelnianie przekraczające granice domeny

Funkcje *Centrum dystrybucji kluczy* są podzielone na dwie odmienne usługi: usługę uwierzytelniania, która wydaje bilety gwarantujące bilet oraz usługę przyznawania biletów (Ticket Granting Service — TGS), która wydaje bilety sesji. Umożliwia to protokołowi *Kerberos* przekraczanie granic domen. Klient może otrzymać bilet przyznający bilet od usługi uwierzytelniania w jednej domenie i użyć go do uzyskania biletów sesji od usługi wydawania biletów z innej domeny.

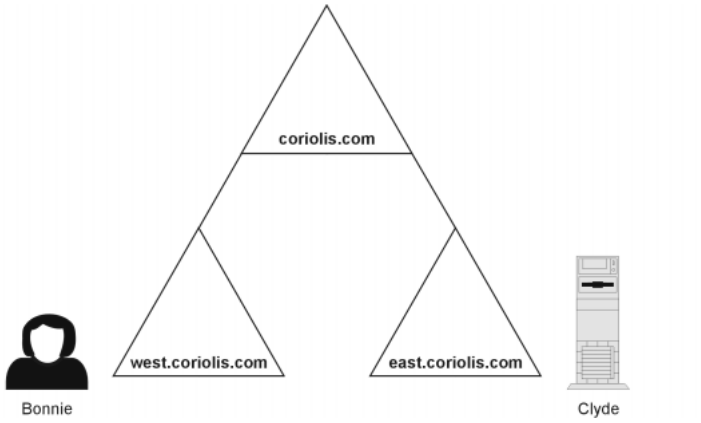
Po pierwsze rozważmy sieć, w której istnieją dwie domeny systemu Windows 2000, *Wschód* i *Zachód*. Jeśli ustanowiono relację zaufania, uwierzytelnianie przekraczające granice domeny zaimplementowano przez współdzielenie klucza międzydomenowego (interdomain key). Usługa przyznawania biletów w każdej z domen jest zarejestrowana jako obiekt typu *security principal* z centrami dystrybucji kluczy innych domen i usługa wydawania biletów w każdej domenie może traktować tę samą usługę w innych domenach jako po prostu jeszcze jedną usługę, dla której klienci, uzyskawszy poprawne uwierzytelnienie, mogą żądać i uzyskać bilety sesji.

Kiedy użytkownik, którego konto znajduje się w domenie *Wschód* chce uzyskać dostęp do serwera, którego konto znajduje się w domenie *Zachód*, proces uwierzytelniania przebiega w następujący sposób:

1. Klient *Kerberos* na stacji roboczej użytkownika wysyła żądanie biletu sesji do usługi wydawania biletów w domenie *Wschód*.
2. Usługa wydawania biletów w domenie *Wschód* widzi, że pożądany serwer nie jest obiektem typu *security principal* w tej domenie i odpowiada, wysyłając klientowi bilet referencyjny, tzn. bilet przyznający bilet, zaszyfrowany za pomocą klucza międzydomenowego, który *Centrum dystrybucji kluczy* w domenie *Wschód* współdzieli z *Centrum dystrybucji kluczy* w domenie Zachód.
3. Klient korzysta z biletu referencyjnego do przygotowania drugiego żądania biletu sesji i tym razem wysyła żądanie do usługi wydawania biletów w domenie *Zachód*.
4. Usługa wydawania biletów w domenie *Zachód* używa swojej kopii klucza międzydomenowego do odszyfrowania biletu referencyjnego. Jeśli bilet referencyjny zostanie odszyfrowany prawidłowo, usługa wydawania biletów wysyła klientowi bilet sesji do żądanego serwera w tej domenie.

Tam, gdzie istnieje drzewo domen, klient z jednej domeny może uzyskać bilet do serwera w innej domenie, przechodząc ścieżką referencyjną przez jedną lub kilka domen pośrednich. Przykład przedstawiono na rysunku 4.7.

Rysunek 4.7. *Uwierzytelnianie poprzez granice domeny*



W firmie, w której pracuje Bonnie, jest domena nadrzędna *helion.com* oraz dwie domeny podrzędne, *wschód.helion.com* i *zachód.helion.com*.

Bonnie loguje się na swoje konto w domenie *zachód.helion.com* i otrzymuje bilet przyznający bilet do *Centrum dystrybucji kluczy* w tej domenie. Bonnie potrzebny jest dostęp do dokumentów zapisanych w udziale publicznym na serwerze w domenie *wschód.helion.com* o nazwie Clyde. Ponieważ domena, w której Bonnie ma swoje konto jest inna niż domena serwera plików, *Dostawca obsługi zabezpieczeń* (SSP) na stacji roboczej Bonnie musi dostać bilet przyznający bilet dla domeny *wschód.helion.com* i użyć go, by uzyskać bilet dla serwera. Proces referencyjny przebiega w następujący sposób:

1. Stacja robocza Bonnie wysyła komunikat *KRB_TGS_REQ* do *Centrum dystrybucji kluczy* w domenie *zachód.helion.com*, który zawiera:
   - nazwę komputera docelowego, Clyde,
   - nazwę domeny, w której znajduje się komputer docelowy, *wschód.helion.com*,
   - bilet przyznający bilet, który daje dostęp do *Centrum dystrybucji kluczy* w domenie *zachód.helion.com*,
   - wartość uwierzytelniającą zaszyfrowaną za pomocą klucza sesji, który Bonnie współdzieli z wyżej wymienionym *Centrum dystrybucji kluczy*.
2. *Centrum dystrybucji kluczy* w domenie *zachód.helion.com* wysyła komunikat *KRB_TGS_REP*. Komunikat ten zawiera:
   - klucz sesji, który Bonnie będzie współdzielić z *Centrum dystrybucji kluczy* z domeny nadrzędnej *helion.com* zaszyfrowany za pomocą klucza sesji logowania Bonnie,
   - bilet przyznający bilet, który daje dostęp do *Centrum dystrybucji kluczy* w domenie *helion.com* zaszyfrowany za pomocą klucza tajnego dla relacji zaufania pomiędzy dwoma domenami *helion.com* i *zachód.helion.com*.
3. Stacja robocza Bonnie wysyła do *Centrum dystrybucji kluczy* w domenie *helion.com* komunikat *KRB_TGS_REQ*, który zawiera:
   - nazwę komputera docelowego, Clyde,
   - nazwę domeny, w której znajduje się komputer docelowy, *wschód.helion.com*,
   - bilet przyznający bilet, który daje dostęp do *Centrum dystrybucji kluczy* w domenie *helion.com*,
   - wartość uwierzytelniającą zaszyfrowaną za pomocą klucza sesji, który Bonnie współdzieli z wyżej wymienionym *Centrum dystrybucji kluczy*.
4. *Centrum dystrybucji kluczy* w domenie *helion.com* wysyła komunikat *KRB_TGS_REP*. Komunikat ten zawiera:
   - bilet przyznający bilet, który daje dostęp do centrum w domenie *wschód.helion.com*, zaszyfrowany za pomocą klucza tajnego dla relacji zaufania pomiędzy dwoma domenami,
   - klucz sesji dla Bonnie, który będzie współdzielony z wymienionym wyżej *Centrum dystrybucji kluczy* zaszyfrowany za pomocą klucza sesji, który Bonnie współdzieli z *Centrum dystrybucji kluczy* w domenie *helion.com*.
5. Stacja robocza Bonnie wysyła do *Centrum dystrybucji kluczy* w domenie *Wschód.helion.com* komunikat *KRB_TGS_REQ*. Komunikat ten zawiera:
   - nazwę komputera docelowego, Clyde,
   - nazwę domeny, w której znajduje się komputer docelowy, *wschód.helion.com*,
   - bilet przyznający bilet, który daje dostęp do *Centrum dystrybucji kluczy* w domenie *wschód.helion.com*,
   - wartość uwierzytelniającą zaszyfrowaną za pomocą klucza sesji, który Bonnie współdzieli z wymienionym wyżej *Centrum dystrybucji kluczy*.

6. *Centrum dystrybucji kluczy* w domenie *wschód.helion.com* wysyła komunikat *KRB_TGS_REP*, który zawiera:
   - klucz sesji dla Bonnie, który będzie współdzielić z komputerem o nazwie Clyde, zaszyfrowany za pomocą klucza sesji współdzielonego przez Bonnie z *Centrum dystrybucji kluczy* w domenie *wschód.helion.com*,
   - bilet sesji, który daje dostęp do komputera o nazwie Clyde, zaszyfrowany za pomocą klucza tajnego, współdzielonego przez Clyde'a z *Centrum dystrybucji kluczy*. Bilet sesji zawiera klucz sesji, który komputer o nazwie Clyde będzie współdzielił z Bonnie, dane autoryzacji, skopiowane z biletu przyznającego bilet Bonnie oraz dane dla domeny lokalnej *wschód.helion.com*.
     Dane autoryzacji zawierają identyfikator zabezpieczeń konta Bonnie, identyfikatory zabezpieczeń dla grup w domenie *zachód.helion.com*, których członkiem jest Bonnie, identyfikatory zabezpieczeń grup uniwersalnych, których członkiem jest Bonnie lub jedna z grup, w której jest członkiem, z domeny *zachód.helion.com* oraz identyfikatory zabezpieczeń dla grup w domenie *wschód.helion.com*, których członkiem jest Bonnie, jedna z jej grup z domeny *zachód.helion.com* lub jedna z jej grup uniwersalnych.
7. Stacja robocza Bonnie wysyła do komputera Clyde komunikat *KRB_AP_REQ*. Komunikat ten zawiera:
   - nazwę główną użytkownika, Bonnie,
   - bilet do komputera o nazwie Clyde,
   - wartość uwierzytelniającą zaszyfrowany za pomocą klucza sesji, który Bonnie współdzieli z komputerem o nazwie Clyde.
8. Komputer Clyde odpowiada komunikatem *KRB_AP_REP*, który zawiera wartość uwierzytelniającą zaszyfrowany za pomocą klucza sesji, który Clyde współdzieli z Bonnie.

Analiza biletów protokołu Kerberos

Co zawierają bilety protokołu *Kerberos*? W jaki sposób oblicza się czas wygaśnięcia (expiration time)? Jak dużą część zawartości biletu jest znana klientowi? Odpowiedzi na te pytania są ważne, jeśli bierze się pod uwagę sposób konfigurowania zasad protokołu *Kerberos*.

Zawartość biletu

W tej części rozdziału wymieniono pola biletu i opisano krótko, jakie informacje zwierają. Dokładne struktury danych biletów, jak również komunikatów można znaleźć w dokumencie *RFC 1510* („The Kerberos Network Authentication Service (V5)") z września 1993, którego autorami są: J. Kohl i C. Neumann. W tabeli 4.1. zamieszczono pierwsze trzy pola biletu. Nie są one zaszyfrowane; informacje zapisane są w postaci zwykłego tekstu, więc klient może je wykorzystać do zarządzania biletami znajdującymi się w buforze.

Tabela 4.1. *Pola biletu zawierające dane zapisane zwykłym tekstem*

Nazwa pola	Opis
Tkt-vno	Numer wersji formatu biletu. W przypadku protokołu Kerberos 5 w polu tym zapisana jest 5
Realm	Nazwa obszaru (realm) — domeny, w której wydano dany bilet. *Centrum dystrybucji kluczy* może wydawać bilety tylko dla serwerów we własnym obszarze, więc jest to także nazwa obszaru serwera
Sname	Nazwa serwera

Pozostałe pola są zaszyfrowane za pomocą klucza tajnego serwera. Pola te przedstawiono w tabeli 4.2.

Tabela 4.2. *Zaszyfrowane pola biletu*

Nazwa pola	Opis
Flags	Opcje biletu
Key	Klucz sesji
Crealm	Nazwa obszaru (domeny) klienta
Cname	Nazwa klienta
Transited	Lista obszarów protokołu Kerberos biorących udział w uwierzytelnianiu klienta, któremu wydano dany bilet
Authtime	Czas pierwszego uwierzytelniania przez klienta. *Centrum dystrybucji kluczy* umieszcza w tym znacznik czasu, gdy wyda bilet przyznający bilet. Kiedy *Centrum dystrybucji kluczy* wydaje bilety na podstawie biletu przyznającego bilet , zapisuje kopię czasu pierwszego uwierzytelnienia biletu przyznającego bilet w polu Authtime w bilecie
Starttime	Czas, po którym bilet jest ważny
Endtime	Data ważności biletu
Renew-till	Maksymalny okres ważności, który można ustawić w bilecie za pomocą znacznika RENEW-ABLE. (opcjonalnie)
Caddr	Jeden lub kilka adresów, z których danych bilet może być wykorzystany. Jeśli pole jest wolne, bilet może być użyty z dowolnego adresu (opcjonalnie)
Authorization-data	Atrybuty uprawnień dla klienta. Protokół Kerberos nie interpretuje zawartości tego pola. Interpretacja jest dokonywana przez usługę (opcjonalnie)

Pole Flags jest polem bitowym, w którym opcje konfiguruje się, ustawiając lub zerując poszczególne bity. Chociaż pole to ma długość 32 bitów, tylko kilka znaczników biletu jest interesujących. Przedstawiono je w tabeli 4.3.

Klienci powinny znać niektóre informacje znajdujące się w biletach i biletach przyznających bilet, aby zarządzać swoim buforem danych uwierzytelniających. *Centrum dystrybucji kluczy*, zwracając bilet i klucz sesji jako wynik wymiany komunikatów usługi uwierzytelniania lub usługi przyznawania biletów, umieszcza kopię klucza sesji klienta w strukturze danych, która zawiera informacje w polach biletu Flags, *Authtime*, *Starttime*, *Endtime* i *Renew-till*. Cała struktura jest zaszyfrowana w kluczu klienta i zwracana za pomocą komunikatów *KRB_AS_REP* i *KRB_TGS_REP*.

Tabela 4.3. *Pole Flags biletu*

Nazwa znacznika	Opis
FORWARDABLE	Informuje usługę przyznawania biletów, że może wydać nowy bilet przyznający bilet z innym adresem sieciowym na podstawie przedstawionego biletu przyznającego bilet (tylko w przypadku biletu przyznającego bilet)
FORWARDED	Wskazuje, że bilet przyznający bilet został przekazany dalej lub, że dany bilet został wydany z przekazanego dalej biletu przyznającego bilet
PROXIABLE	Informuje usługę przyznawania biletów, że może wydawać bilety z innymi adresem sieciowym niż ten, który znajduje się w danym bilecie przyznającym bilet (tylko w przypadku biletu przyznającego bilet)
PROXY	Wskazuje, że adres sieciowy w bilecie jest inny niż adres w bilecie przyznającym bilet użytym do uzyskania danego biletu
RENEWABLE	Używany w kombinacji z polami Endtime i Renew-till, aby bilety z długim okresem ważności były odświeżane okresowo w Centrum dystrybucji kluczy
INITIAL	Wskazuje, że jest to bilet przyznający bilet (tylko w przypadku biletu przyznającego bilet)

Ograniczanie czasu życia biletu

Bilety protokołu *Kerberos* mają określony czas początkowy i czas ważności. W okresie ważności biletu klient, który otrzymał ten bilet do realizacji pewnej usługi, może go przedstawić i uzyskać dostęp do tej usługi, bez względu na to, ile razy korzystał wcześniej z danego biletu. Aby ograniczyć zagrożenia dla bezpieczeństwa biletu lub odpowiadającemu mu klucza sesji, administratorzy mogą ustawić maksymalny czas „życia" biletów.

Kiedy klient żąda od *Centrum dystrybucji kluczy* biletu do jakiejś usługi, może żądać określonego czasu początkowego. Jeśli żądanie nie zawiera tego czasu lub on minął, *Centrum dystrybucji kluczy* wpisuje czas aktualny w polu *Starttime* biletu.

Niezależnie od tego, czy klienci podają czasy początkowe ich żądania muszą zawierać wymagany czas ważności. *Centrum dystrybucji kluczy* ustala wartość zapisaną w polu *Endtime* danego biletu, dodając maksymalną wartość okresu ważności biletu, podaną w zasadach protokołu *Kerberos*, do wartości zapisanej w polu *Starttime* tego biletu a następnie porównuje wynik z wymaganym czasem ważności biletu. Zostanie ten czas, który jest wcześniejszy.

Centrum dystrybucji kluczy nie powiadamia klientów, kiedy kończy się okres ważności biletów sesji lub biletów przyznających bilet. Jeśli klient, żądając połączenia z serwerem, przedstawia bilet sesji, który utracił ważność, to serwer odsyła komunikat o błędzie i klient musi uzyskać nowy bilet sesji z *Centrum dystrybucji kluczy*. Jednak jeśli połączenie zostało uwierzytelnione, to nie ma znaczenia czy bilet sesji traci ważność w tej sesji. Bilety sesji używane są tylko do uwierzytelniania nowych połączeń z serwerem i operacje w toku nie są przerywane w przypadku utraty ważności biletu sesji.

W przypadku, gdy klient, żądając od *Centrum dystrybucji kluczy* biletu sesji, przedstawi bilet przyznający bilet, który utracił ważność, wówczas centrum (KDC) odpowiada komunikatem o błędzie. Klient musi zażądać nowego biletu przyznającego bilet , jak i potrzebny mu będzie klucz długoterminowy użytkownika. Jeśli podczas wstępnego logowania klient nie zapisał tego klucza w buforze, to może zażądać hasła użytkownika.

Odnawialne bilety protokołu Kerberos

Jednym ze sposobów obrony przeciwko „atakom na klucze sesji" jest takie ustanowienie zasad protokołu *Kerberos*, aby maksymalny okres ważności biletu był stosunkowo krótki. Innym ze sposobów obrony jest umożliwienie stosowania biletów odnawialnych. Gdy bilety są odnawialne, to klucze sesji są odświeżane okresowo bez wydawania zupełnie nowego biletu. Jeśli zasady protokołu *Kerberos* zezwalają na stosowanie biletów odnawialnych, to *Centrum dystrybucji kluczy* ustawia znacznik *RENEWABLE* w każdym bilecie, który wydaje i ustawia w nim dwa czasy ważności. Pierwszy czas ogranicza okres ważności bieżącej postaci danego biletu, a drugi ogranicza łączny okres ważności wszystkich postaci danego biletu.

Czas ważności bieżącej postaci danego biletu jest zapisany w polu *Endtime*. Klient, który ma bilet odnawialny musi wysłać go do *Centrum dystrybucji kluczy*, aby został odnowiony, zanim osiągnięty zostanie czas zapisany w polu *Endtime*, przedstawiając także nową wartość uwierzytelniającą. Kiedy centrum otrzyma bilet do odnowienia, sprawdza drugi, łączny czas ważności zapisany w polu *Renew-till* i upewnia się, czy ten czas już nie upłynął. Następnie wydaje nową postać biletu z późniejszym czasem ważności *Endtime* i nowym kluczem sesji. Oznacza to, że administrator może ustalić zasady protokołu *Kerberos*, tak aby bilety musiały być odnawiane w stosunkowo krótkich okresach czasu, ponieważ w takich sytuacjach wydawany jest nowy klucz sesji, co minimalizuje straty, które mogłyby powstać w wyniku zagrożeń bezpieczeństwa klucza. Administratorzy mogą również podać stosunkowo długi łączny czas ważności biletu, po upływie którego bilet traci ostatecznie ważność i nie podlega odnowieniu.

Delegowanie uwierzytelniania

W wielowarstwowych aplikacjach klient-serwer, klient łączy się z serwerem, który z kolei musi połączyć się z drugim, wewnętrznym serwerem. Aby do tego doszło, pierwszy serwer musi mieć bilet do drugiego. Byłoby doskonale, gdyby bilet ograniczał prawo dostępu pierwszego serwera do drugiego w sytuacjach, w których klient, a nie pierwszy serwer, jest upoważniony.

W protokole *Kerberos* rozwiązano ten problem za pomocą mechanizmu zwanego *Delegowaniem uwierzytelniania* (Delegation of Authentication), który polega na delegowaniu przez klienta uwierzytelnienia do serwera poprzez poinformowanie *Centrum dystrybucji kluczy*, że serwer jest uprawniony do reprezentowania tego klienta. Jest to idea podobna do personifikacji występującej w systemie Windows 2000.

Delegowania można dokonać na dwa sposoby:

- *bilety pośrednie* (proxy tickets) — klient otrzymuje bilet dla serwera wewnętrznego, a następnie przekazuje go do serwera zewnętrznego. Trudność stosowania tego sposobu polega na tym, że klient musi znać nazwę serwera wewnętrznego,
- *bilety przekazywane* (forwarded tickets) — klient daje serwerowi zewnętrznemu bilet przyznający bilet, który serwer może w razie potrzeby używać do żądania biletów.

Zasady protokołu *Kerberos* domeny mogą korzystać tylko z jednej z tych metod.

Bilety pośrednie

Kiedy *Centrum dystrybucji kluczy* wydaje klientowi bilet przyznający bilet, sprawdza, czy zasady protokołu *Kerberos* dopuszczają używanie biletów. Jeśli tak jest, *Centrum dystrybucji kluczy* ustawia znacznik *PROXIABLE* w wydanym danemu klientowi bilecie gwarantującym bilet.

Klient uzyskuje bilet pośredni, przedstawiając bilet przyznający bilet usłudze przyznawania biletów i żądając biletu do serwera wewnętrznego. Żądanie wysłane przez klienta zawiera znacznik, który sygnalizuje, że konieczny jest bilet pośredni, a także nazwę serwera, który będzie reprezentował tego klienta. Kiedy centrum (KDC) otrzymuje żądanie klienta, tworzy bilet dla serwera wewnętrznego, ustawia w tym bilecie znacznik *PROXY* i wysyła z powrotem do klienta. Następnie klient wysyła bilet do serwera zewnętrznego, który korzysta z tego biletu, aby uzyskać dostęp do serwera wewnętrznego.

Bilety przekazywane

Jeśli klient chce delegować do serwera zewnętrznego zadanie uzyskiwania biletów dla serwera wewnętrznego, to żąda od *Centrum dystrybucji kluczy* przekazywalnego biletu przyznającego bilet. Klient robi to za pomocą komunikatu żądania podprotokołu *Authentication Service Exchange*, wskazując *Centrum dystrybucji kluczy* nazwę serwera, który będzie działał w jego imieniu. Jeśli zasady protokołu *Kerberos* zezwalają na przekazywanie, to *Centrum dystrybucji kluczy* tworzy bilet przyznający bilet dla serwera zewnętrznego, który jest używany w imieniu klienta, ustawia znacznik *FORWARDABLE* i wysyła bilet przyznający bilet z powrotem do klienta. Następnie klient przekazuje ten bilet do serwera zewnętrznego.

Serwer zewnętrzny, żądając biletu do serwera wewnętrznego przedstawia *Centrum dystrybucji kluczy* bilet przyznający bilet klienta. Centrum, wydając bilet, sprawdza znacznik *FORWARDABLE* w tym bilecie przyznającym bilet, ustawia w wydawanym bilecie znacznik *FORWARDED* i zwraca bilet do serwera zewnętrznego.

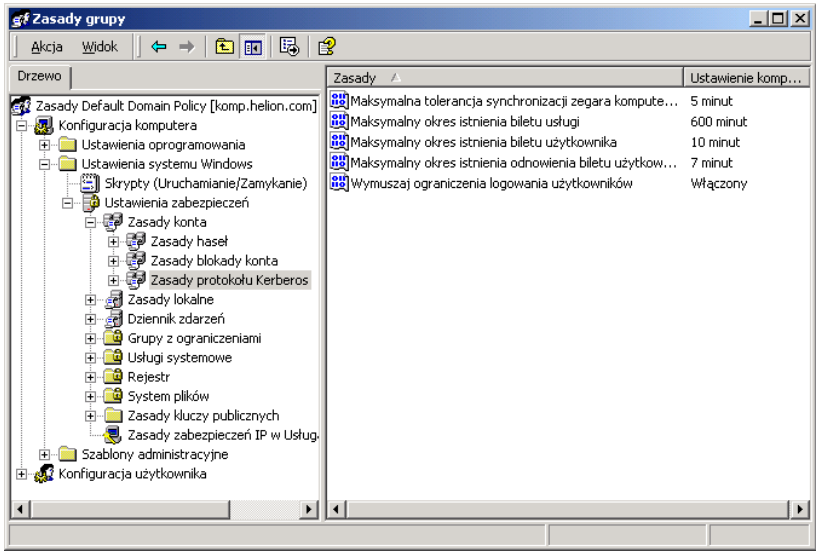
Konfigurowanie zasad protokołu Kerberos domeny

W systemie Windows 2000 zasady protokołu *Kerberos* są określane na poziomie domeny i wprowadzane w życie przez *Centrum dystrybucji kluczy* danej domeny. Zasady protokołu *Kerberos* są zapisane w *Active Directory* jako podzbiór atrybutów zasad zabezpieczeń domeny. Domyślnie zasady mogą być konfigurowane tylko przez administratorów domeny. Do ustawień zasad protokołu *Kerberos* można uzyskać dostęp: edytując obiekt zasad grup *Zasady domeny* (Domain Policies), rozwijając gałęzie *Konfiguracja komputera* (Computer Configuration), *Ustawienia systemu Windows* (Windows Settings), *Ustawienia zabezpieczeń* (Security Settings) i *Zasady konta* (account policies), a następnie wybierając pozycję *Zasady protokołu Kerberos* (Kerberos Policies), jak przedstawiono to na rysunku 4.8.

Zasady protokołu Kerberos obejmują następujące ustawienia:

- *Wymuszaj ograniczenia logowania użytkowników* (Enforce user logon restrictions) — jeśli ta opcja jest ustawiona, to *Centrum dystrybucji kluczy* zatwierdza każde żądanie biletu sesji sprawdzając zasady praw użytkownika na komputerze

Rysunek 4.8. *Ustawienia zasad protokołu Kerberos domeny*



docelowym, aby potwierdzić, że użytkownik ma prawo albo do logowania lokalnego, albo do dostępu poprzez sieć. Weryfikacja jest opcjonalna, ponieważ dodatkowy etap jest czasochłonny i może spowolnić dostęp sieciowy do usług. Domyślnie ustawione jest *Włączony* (Enabled),

- *Maksymalny okres istnienia biletu usługi* (Maximum lifetime for service ticket) — określenie bilet usługi oznacza bilet sesji. Okres ważności podany jest w minutach i musi być dłuższy niż 10 minut i mniejszy lub równy wartości *Maksymalny czas istnienia biletu użytkownika*. Domyślną wartością jest 600 minut (10 godzin),
- *Maksymalny czas istnienia biletu użytkownika* (Maximum lifetime for user ticket) — określenie bilet użytkownika oznacza bilet przyznający bilet. Wartość podana jest w godzinach (domyślną wartością jest 10 godzin),
- *Maksymalny czas istnienia odnowienia biletu użytkownika* (Maximum lifetime for user ticket renewal) — wartość podana jest w dniach (domyślnie jest to 7 dni),
- *Maksymalna tolerancja synchronizacji zegara komputerowego* (Maximum tolerance for computer clock synchronization) — wartość podana jest w minutach (domyślnie — 5 minut).

Dostawca obsługi zabezpieczeń Kerberos

Dostawca obsługi zabezpieczeń (Security Support Provider — SSP) był wspominany w niniejszym rozdziale kilkakrotnie, ale bez szczegółowych wyjaśnień. *Dostawca obsługi zabezpieczeń* jest to biblioteka DLL dostarczana z systemem operacyjnym, która jest implementacją protokołu zabezpieczeń *Kerberos*. System Windows 2000 zawiera także dostawcę obsługi zabezpieczeń (SSP) dla uwierzytelniania za pomocą protokołu *NTLM*. Obydwie biblioteki są domyślnie pobierane podczas rozruchu systemu przez *Lokalne źródło zabezpieczeń* (LSA), pracującą na komputerze z systemem Windows 2000. Każdy

z dostawców może być użyty do uwierzytelniania logowania sieciowego i połączeń klient-serwer. To, który zostanie zastosowany, zależy od możliwości komputera po drugiej stronie połączenia. Jako pierwszy wybierany jest dostawca zabezpieczeń *Kerberos*.

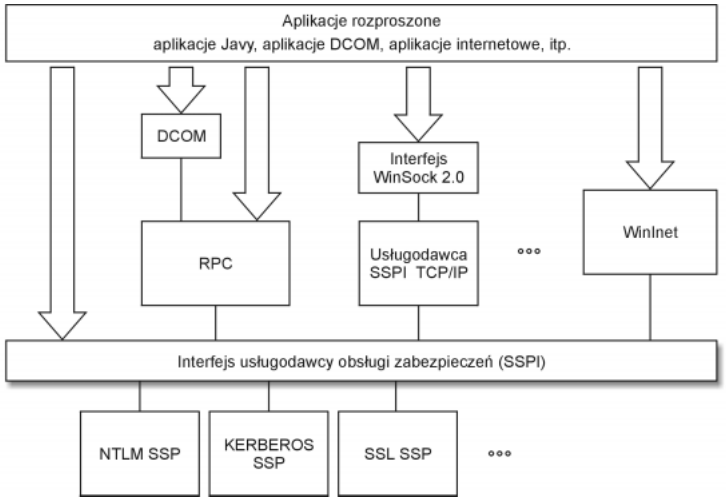
Po tym, jak *Lokalne źródło zabezpieczeń* ustanowi kontekst zabezpieczeń dla użytkownika interaktywnego, proces obsługi podpisywania i pieczętowania komunikatów, działający w kontekście zabezpieczeń użytkownika, może załadować kolejną kopię) *Dostawcy obsługi zabezpieczeń Kerberos*.

Usługi systemowe i aplikacje warstwy transportowej mają dostęp do dostawców obsługi zabezpieczeń za pomocą *Interfejsu Dostawcy obsługi zabezpieczeń* (Security Support Provider Interface — SSPI). Jest to interfejs *Win32*, który służy do wyliczania dostawców dostępnych w systemie, wyboru jednego z nich i użycia go w celu uzyskania połączenia uwierzytelnionego. Interfejs *SSPI* omówiono poniżej.

Zastosowanie interfejsu dostawcy obsługi zabezpieczeń

Interfejs Dostawcy obsługi zabezpieczeń (SSPI) jest interfejsem *Win32* pomiędzy aplikacjami warstwy transportowej a dostawcami zabezpieczeń sieciowych. Integruje on uwierzytelnianie, spójność i poufność komunikatów z aplikacjami rozproszonymi. Szkielet aplikacji modelu obiektowego składników rozproszonych (DCOM) i uwierzytelnione, zdalne wywołania procedur (authenticated Remote Procedure Calls) korzystają z usług *Interfejsu Dostawcy obsługi zabezpieczeń* interfejsów wyższego poziomu. Usługi zabezpieczeń interfejsu *SSPI* są również zintegrowane z interfejsami poziomu aplikacji takimi jak *WinSock 2.0* i *WinInet*.

Na rysunku 4.9 przedstawiono miejsce usług zabezpieczeń interfejsu *SSPI* w całej strukturze aplikacji rozproszonej.



Rysunek 4.9. *Interfejs dostawcy obsługi zabezpieczeń i model bezpieczeństwa systemu Windows*

Interfejs Dostawcy obsługi zabezpieczeń stanowi warstwę abstrakcji pomiędzy protokołami poziomu aplikacji a protokołami zabezpieczeń. Usługi tego interfejsu *SSPI* mogą być używane w różny sposob:

1. Tradycyjne aplikacje oparte na gniazdach (socket based applications) mogą wywoływać procedury *Interfejsu Dostawcy obsługi zabezpieczeń* bezpośrednio i zastosować protokół aplikacji, który przekazuje dane dotyczące bezpieczeństwa *Interfejsu Dostawcy obsługi zabezpieczeń* za pomocą komunikatów żądania i odpowiedzi.
2. Aplikacje mogą korzystać z modelu DCOM do wywoływania opcji zabezpieczeń, które zaimplementowano za pomocą uwierzytelnionych zdalnych wywołań procedur i *Interfejsu Dostawcy obsługi zabezpieczeń* na niższych poziomach. Aplikacje nie wywołują bezpośrednio tego interfejsu.
3. *WinSock 2.0* rozszerza interfejs gniazd systemu Windows (Windows Sockets interface), aby umożliwić usługodawcom transportu ujawnienie funkcji zabezpieczeń. Takie podejście integruje *Interfejs Dostawcy obsługi zabezpieczeń* ze stosem sieciowym i stanowi wspólny interfejs dla usług zabezpieczeń i transportowych.
4. *WinInet* jest interfejsem protokołu aplikacji, który jest przeznaczony do obsługi zabezpieczonych protokołów internetowych, takich jak *Secure Socket Layer* (SSL). Obsługa zabezpieczeń WinInet stosuje *Interfejs Dostawcy obsługi zabezpieczeń* do dostawcy zabezpieczeń bezpiecznego kanału.

Języki skryptowe, takie jak *VBscript* czy *JScript* zezwalają programistom na dostęp do *Interfejsu Dostawcy obsługi zabezpieczeń*. O ile tworzenie aplikacji wykracza poza zakres niniejszej książki, to administrator zabezpieczeń powinien mimo wszystko wiedzieć o strukturze interfejsu dostawcy obsługi zabezpieczeń i jego miejscu w architekturze systemu Windows 2000.

Interfejs Dostawcy obsługi zabezpieczeń stanowi wspólny interfejs pomiędzy aplikacjami warstwy transportowej, takimi jak *Microsoft RPC* lub program przeadresowujący systemu plików oraz dostawcami zabezpieczeń. Dostarcza również mechanizmy, za pomocą których aplikacje rozproszone mogą wywoływać jednego z wielu usługodawców zabezpieczeń, aby uzyskać połączenie uwierzytelnione, bez znajomości szczegółów protokołu zabezpieczeń.

Interfejs Dostawcy obsługi zabezpieczeń składa się z czterech rodzajów interfejsów:

1. *Interfejsy do zarządzania danymi uwierzytelniającymi* (Credential management interfaces) — umożliwiają uzyskanie dostępu do danych uwierzytelniających — dane, hasła, bilety, itd. — lub zwolnienie tego dostępu. Dostępne są następujące metody:
   - *AcquireCredentialsHandle* — uzyskuje uchwyt do wskazanych danych uwierzytelniających,
   - *FreeCredentialsHandle* — zwalnia uchwyt do danych uwierzytelniających i związane z tym zasoby,
   - *QueryCredentialsAttributes* — umożliwiają wysyłanie zapytań dotyczących różnych atrybutów danych uwierzytelniających, takich jak skojarzona nazwa, nazwa domeny, itd.

2. *Interfejsy do zarządzania kontekstem* (Context management interfaces) — dostarczają metody do tworzenia i stosowania kontekstów zabezpieczeń. Konteksty te są tworzone zarówno po stronie klienta, jak i serwera połączenia komunikacyjnego i mogą być używane później z interfejsami obsługi komunikatów. Dostępne są następujące metody:
   - *InitializeSecurityContext* — inicjuje kontekst zabezpieczeń, generując komunikat nieprzejrzysty (opaque message) — znacznik zabezpieczeń — który może być przekazany do serwera,
   - *AcceptSecurityContext* — tworzy kontekst zabezpieczeń, za pomocą komunikatu nieprzejrzystego otrzymanego od klienta,
   - *DeleteSecurityContext* — zwalnia kontekst zabezpieczeń i związane z nim zasoby,
   - *QueryContextAttributes* — pozwala na wysyłanie zapytań dotyczących różnych atrybutów kontekstu,
   - *ApplyControlToken* — wprowadza dodatkowe komunikaty zabezpieczeń do istniejącego kontekstu zabezpieczeń,
   - *CompleteAuthToken* — uzupełnia znacznik uwierzytelnienia. Jest to konieczne, ponieważ w przypadku niektórych protokołów, takich jak *Distributed Computing Environment Remote Procedure Call* (DCE RPC), konieczne jest przejrzenie informacji o zabezpieczeniach, gdy tylko transport uaktualni określone pola komunikatu,
   - *ImpersonateSecurityContext* — dołącza kontekst zabezpieczeń klienta do wywoływanego wątku jako znacznik personifikacji,
   - *RevertSecurityContext* — przerywa personifikację i przywraca wywoływanemu wątkowi znacznik pierwotny.
3. *Interfejsy obsługi komunikatów* (Message support interfaces) — zawierają usługi zapewniające spójność i poufność komunikacji, które są oparte na kontekście zabezpieczeń. Dostępne są następujące metody:
   - *MakeSignature* — generuje podpis zabezpieczony na podstawie komunikatu i kontekstu zabezpieczeń,
   - *VerifySignature* — sprawdza, czy podpis jest zgodny z odebranym komunikatem.
4. *Interfejsy do zarządzania pakietami* (Package- management interfaces) — zawierają usługi dla różnych pakietów zabezpieczeń obsługiwanych przez dostawcę zabezpieczeń. Dostępne są następujące metody:
   - *EnumerateSecurityPackages* — przedstawia listę dostępnych pakietów zabezpieczeń i ich możliwości,
   - *QuerySecurityPackageInfo* — wysyła do pojedynczych pakietów zabezpieczeń zapytania dotyczące ich możliwości.

Dostawca zabezpieczeń to biblioteka DLL, która jest implementacją interfejsu dostawcy obsługi zabezpieczeń i udostępnia aplikacjom jeden lub kilka pakietów zabezpieczeń. Pakiet zabezpieczeń *SSP* przyporządkowuje funkcje interfejsu *SSPI* implementacji protokołu zabezpieczeń, właściwej dla danego pakietu, takiego jak *NTLM*, *Kerberos* czy *SSL*. Na etapie inicjalizacji identyfikacji określonego pakietu używana jest nazwa pakietu zabezpieczeń.

Interfejs *SSPI* pozwala aplikacji na używanie dowolnego, dostępnego w systemie pakietu zabezpieczeń, bez zmiany interfejsu do korzystania z usług zabezpieczeń. Ponadto nie ustanawia danych uwierzytelniających logowania, ponieważ zazwyczaj jest to operacja uprzywilejowana, przeprowadzana przez system operacyjny.

Aplikacja może użyć funkcji zarządzania pakietami do prezentowania listy dostępnych pakietów zabezpieczeń i wybrania tego, który jest potrzebny. Następnie aplikacja korzysta z funkcji do zarządzania danymi uwierzytelniającymi, aby uzyskać uchwyt do danych uwierzytelniających użytkownika, w którego imieniu działa. Mając to dojście, aplikacja może skorzystać z funkcji do zarządzania kontekstem, aby utworzyć kontekst zabezpieczeń usługi. Kontekst zabezpieczeń jest to struktura danych nieprzejrzystych zawierająca dane o zabezpieczeniach odpowiednich dla połączenia, takiego jak klucz sesji, czas trwania sesji, itd. Ostatecznie aplikacja korzysta z kontekstu zabezpieczeń z funkcjami obsługi komunikatów dla zapewnienia integralności i poufności komunikacji podczas trwania połączenia.

Możliwości pakietów zabezpieczeń

Możliwości pakietu zabezpieczeń określają, jakie usługi dostarcza on danej aplikacji. Obejmują one, np. obsługę uwierzytelniania klienta uwierzytelnianie wzajemne lub obsługę spójności komunikatów oraz ich poufności. Dodatkowo niektóre pakiety są przeznaczone do korzystania tylko z niezawodnych protokołów transportowych, a nie z transportów datagramowych.

Możliwości pakietów zabezpieczeń, dostępne w określonym pakiecie, można uzyskać za pomocą funkcji *QuerySecurityPackageInfo*. Poniższa lista przedstawia możliwości pakietów zabezpieczeń.

1. Możliwości związane z uwierzytelnianiem:
   - uwierzytelnianie wyłącznie klienta,
   - wymagane uwierzytelnianie wieloetapowe,
   - obsługa personifikacji w systemie Windows.
2. Możliwości związane z transportem:
   - transport z wykorzystaniem datagramów,
   - transporty połączeniowe,
   - semantyka połączenia strumienia danych.
3. Możliwości związane z komunikatami:
   - obsługa integralności komunikatów,
   - obsługa poufności komunikatów.

W większości przypadków aplikacje wybierają pakiety zabezpieczeń na podstawie rodzaju dostępnych możliwości zabezpieczeń, które zaspokajają potrzeby danej aplikacji.

Powyższe informacje na temat interfejsu *SSPI* dają jego pełny obraz, bez wnikania w szczegóły programowania. Więcej wiadomości na ten temat można znaleźć w dokumentacji systemu Windows 2000: *Books OnLine* oraz *Technet*. Jeśli ktoś chce spróbować samodzielnego programowania, najpierw niech zapozna się z przykładami zamieszczonymi w *Windows 2000 Software Development Kit* (SDK).