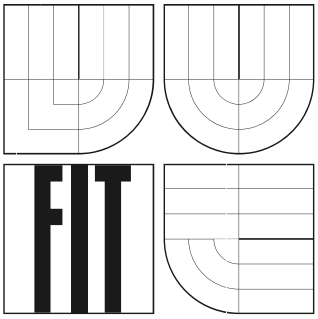
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Single sign-on v J2EE webových aplikacích

Semestrální projekt

2008 **Tomáš Nečas**

Single sign-on v J2EE webových aplikacích

Tuto semestrální práci jsem vypracoval samostatně pod vedením

Ing. Ondřeje Ryšavého, Ph.D.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Zadání:

- Analyzujte možná řešení single sign-on pro webové aplikace založená na J2EE aplikačním serveru.
- Porovnejte různé autentizační protokoly (Kerberos, NTLM).
- Popište úroveň podpory autentizace v prohlížečích (IE, Firefox, Opera) na různých OS (Windows, Linux).
- Popište úroveň podpory autentizace ve webových serverech (Apache, Java aplikační servery, IIS).

Na semestrální projekt bude navazovat diplomová práce, která jej doplní o následující body:

- Naprogramujte v jazyku Java podporu na straně J2EE serveru pro autentizaci doménového uživatele windows do webové aplikace pomocí protokolu SPNEGO/Kerberos.
- Zhodnoťte dosažené výsledky.

Abstrakt

Semestrální práce se zabývá potřebou, analýzou, porovnáním a popisem možných řešení Single Sign-On pro webové aplikace založená na J2EE aplikačním serveru.

Práce je zaměřena především na protokoly Kerberos a NTLM. Jedná se o rozdílné přístupy k řešení jednotného přihlášení, proto je také analýza zaměřena na architekturu systémů, které Single Sign-On implementují. Práce přináší přehled o základních principech a konceptech SSO. U jednotlivých konceptů je diskutována jejich bezpečnost, proces autentizace a využití v rámci dnešních webových aplikací. Dále jsou představeny konkrétní implementace a řešení protokolů na J2EE platformě. Dokument představuje hlavní trendy a motivaci pro implementaci Single Sign-On v oblasti podnikové sféry.

Klíčová slova

Single Sign-On, Kerberos, NTLM, J2EE, CIFS, SMB, autentizace, autorizace

Obsah

1 Úvod

Pro uživatele počítače není lehké zapamatovat si množství autentizačních údajů pro přístup k nejrůznějším službám systému. Přihlašování se stává složitou záležitostí zvláště když je třeba, abychom si zapamatovali pro každý systém nebo zdroj jiné uživatelské jméno a heslo. Heslo, které je jednoduše zapamatovatelné, lze zase snadněji uhodnout a pro změnu opakování jednoho hesla pro více služeb se zvyšuje riziko jeho odhalení. Bezpečné není ani poznamenávání si hesla na papír nebo dokonce do počítače, kde by je mohl útočník najít a zneužít. A k tomu všemu se často setkáváme s požadavkem, abychom svá přístupová hesla často měnili.

Je tedy zřejmé, že IT manažeři přijímají v posledních letech myšlenku jednotného přihlašování, díky kterému lze podstatně snížit náklady na správu a podporu uživatelů. Díky řešení jednotného přihlašování budou uživatelé disponovat pouze jediným heslem a přitom se nesníží zabezpečení přístupu různých služeb v rámci sítě [10].

Jednotné přihlašování Single Sign-on (dále SSO) nabízí řešení, které zjednoduší život jak uživatelům, tak administrátorovi.

1.1 Obsah práce

Práce je napsána tak, aby poskytla čtenáři obecný přehled o principech a možnostech, které poskytuje Single Sign-on. V prvním kroku se tedy jedná o teoretickou část, na kterou bude navazovat část praktická, která bude řešit implementaci protokolu Kerberos na J2EE aplikačním serveru.

Text je členěn na kapitoly tak, aby čtenář dostal ucelený obraz problematiky. Následuje přiblížení obsahu jednotlivých kapitol.

Potřeba Single Sign-on (kapitola 2)

Kapitola se zabývá základní charakteristikou architektur webových aplikací s ohledem na autentizaci a bezpečnost. Na základě tohoto přehledu nachází potřebu a motivaci k řešení SSO.

Řešení Single Sign-on (kapitola 3)

Vše podstatné o SSO. Zabývá se definicí SSO, základními principy a koncepty SSO. Součástí kapitoly jsou také ukázky architektur a modelů, které řeší a implementují SSO.

Autentizační protokoly (kapitola 4)

Kapitola představuje základní charakteristiku protokolů NTLM a Kerberos. Dále se zabývá porovnáním základních rysů těchto protokolů

NTLM (kapitola 5)

Kapitola představuje základy autentizačního protokolu NTLM. Součástí je vysvětlení procesu autentizace a možnosti implementace protokolu NTLM.

Kerberos (kapitola 6)

Kapitola vysvětluje základní principy protokolu Kerberos a zabývá se podrobněji popisem komunikace, která vede k ověření přístupu ke zdroji. Součástí kapitoly je výčet implementací protokolu.

Aplikační podpora protokolů (kapitola 7)

Kapitola se zabývá úrovní podpory protokolů NTLM a Kerberos jak na straně klienta, tak na straně serveru.

Práce je souhrnem informací, které získal autor při řešení různých úloh a implementací, které se týkaly Single Sign-on.

1.2 Pojmy

Autentizace – proces ověření identity.

Autorizace – postup, který omezuje přístup k informacím, funkcím a dalším objektům. Přístup mají pouze oprávněné subjekty.

Kerberos – síťový autentizační protokol vytvořený na MIT (Massachusetts Institute of Technology) a primárně určený pro klient-server model.

NTLM – (zkratka z NT LAN Manager) autentizační protokol, který je součástí konceptu Windows autentizace.

SPNEGO – Negotiate – protokol, který určuje dostupný mechanismus pro autentizaci (NTLM, Kerberos).

Basic Auth – jednoduchá implementace autentizažního mechanismu se zobrazením výzvy k zadaní uživatelského jména a hesla.

jCIFS – servlet filter, který autentizuje pomocí SMB protokolu a využívající Windows credentials.

Architektura klient – server – síťová architektura, která odděluje klienta a server. Jednotlivé instance klientů komunikují se serverem, který obvykle běží na vzdáleném počítači

WEBACS – vícevrstvá architektura client – server s datovou logikou a Java method serverem na straně Content Serveru. Zkratka je vymyšlena autorem kvůli častému odkazovaní na tento typ architektury

Domain Controller – (doménový řadič) server v síti Windows NT, na němž jsou uloženy všechny informace pro správu systému

Doména - základní adresní jednotkou na Internetu. Překladem domény je IP adresa

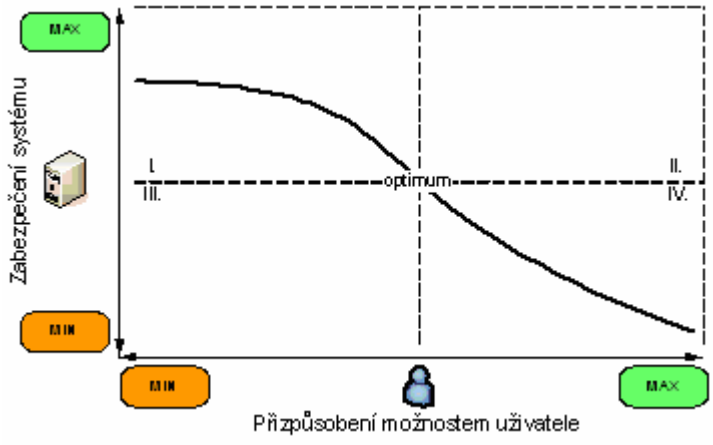
Policy Server – bezpečnostní komponenta v síti, která poskytuje autorizační služby

2 Potřeba Single Sign-on

2.1 Motivace

Podniková sféra velkých společností nejrůznějších oborů, mezi které patří mimo jiné bankovní sektor, automobilový a farmaceutický průmysl, využívá ve většině případů informační systémy jako základ úspěšného běhu podniku. Do tohoto základu patří efektivní komunikace, přehledná úložiště, možnosti auditu, výkonné operace a další úkony a stavy informačních systémů. S těmito systémy pak pracuje velké množství lidí. Proto je v rámci oboru zjišťování efektivnosti pracovních sil kladen nemalý důraz na efektivní využití informačních systémů. Jako příklad si můžeme vzít firmu, která prodává hračky. Údaje o každé hračce se v této firmě zavádí skrze informační systém do databáze. Vzhledem k tomu, že těchto operací udělá jeden uživatel během dne řádově stovky, je třeba při návrhu takového systému počítat s maximální optimalizací procesu vkládání hračky do báze, jinak dojde k vysokým ztrátám efektivnosti lidského zdroje.V reálném provozu však k rutinnímu úkolu vkládání hraček do systému připadne ještě zapisování odpracovaných hodin, čtení doručené pošty a například přístup na informační portál. Všechny tyto aktivity vyžadují přihlášení k patřičnému serveru, který poskytuje danou (webovou) službu. A zde se dostáváme k činnosti, která denně mnohokrát opakuje a přitom která je prioritní z hlediska bezpečnosti dat osobních i celé firmy.

Hlavní potřebou podnikové sféry v oblasti autentizace uživatelů, ze které pak následně vzešel přístup nazvaný Single Sign-on, je tedy maximalizovat bezpečnost a minimalizovat časové a intelektuální náklady uživatele podnikových systémů. Jak je však patrné, existuje zde nepřímá úměra.



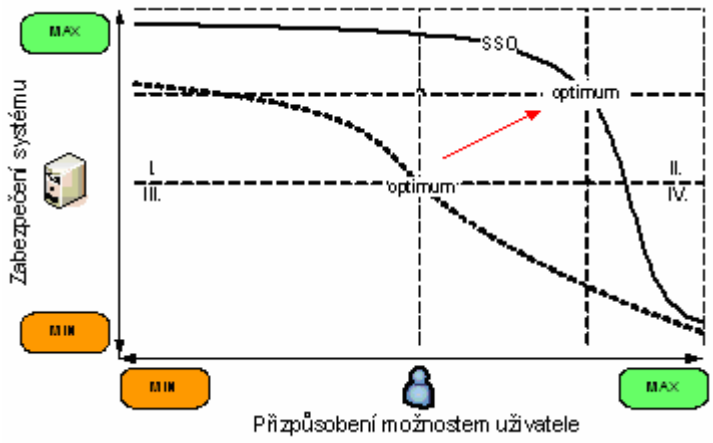
Obrázek 1: Graf závislosti zabezpečení systému na uživatelových možnostech

Pokud je snaha o co nejbezpečnější intranetovou síť, pak se tak děje na úkor intelektuálních možností uživatelů. A naopak, pokud se snažíme, aby přístup byl pro uživatele řešen co nejjednodušeji – mj.

jednoduchá a lehce zapamatovatelná hesla, stejné přístupové údaje ke všem službám intranetu – ,pak se tak děje vždy na úkor bezpečnosti sítě. Tento vztah je vyznačen křivkou v diagramu (Obrázek 1).

Jak je na diagramu vidět, optimálním řešením je bod optimum, který definuje vyváženou variantu poměru bezpečnosti a uživatelských možností. Když si všimneme křivky v I. kvadrantu, má poněkud jiný tvar od křivky vedoucí ve IV. kvadrantu. To je způsobeno způsobem řešení bezpečnosti z pohledu uživatele. Pokud je v rámci firmy vedena politika bezpečnosti na minimální úrovni, kde například po zapnutí firemního počítače a přihlášení do místní sítě již uživatel není žádán k potvrzení hesla do žádné aplikace, pak je to z hlediska uživatele maximálně výhodné (ještě pokud je heslem uživatele jeho jméno), nicméně z hlediska bezpečnosti se jedná o minimální a nedostačující řešení. Opačným řešením je pak maximalizace bezpečnosti. To je možné představit si zadáváním složitých, dlouhých a různorodých hesel do každé z aplikací. V tomto případě se jedná z pohledu uživatele o velmi nesnadný úkol, který není možné ustát bez toho, aniž by hesla nebyla někde poznačena, uložena nebo jinak připomenuta. S tímto faktem ale zase vzrůstá možnost nalezení identifikačních údajů útočníkem a možnost napadnutí systému pod cizí identitou. To stejné platí i pro řešení různých PKI a jiných klíčů. Ač je toto řešení vysoce účinné a bezpečné, vždy je určitá pravděpodobnost, že bude klíč uživateli odcizen a použit k přístupu do systému. Proto v rámci klasického způsobu přihlašování není docíleno maximální bezpečnosti.

Hlavní potřeba tedy spočívá ve zvýšení bezpečnosti a snížení nároků na uživatele v multi-aplikačním prostředí. Tuto potřebu splňuje řešení Single Sign-on.



Obrázek 2: Řešení Single Sign-On v grafu závislosti zabezpečení systému na uživatelových možnostech

Na obrázku (Obrázek 2) je vidět posun polohy křivky v rámci diagramu. Křivka se dostává i do II. kvadrantu a obecně se dá říci, že bod optima pro nastavení správného poměru bezpečnosti a uživatelských možností se nachází v bodě, které je charakterizováno jednak vysokým standardem bezpečnostní politiky v rámci firemní sítě, tak vstřícným přístupem k uživateli a jeho možnostem jak

časovým tak intelektuálním. Toho je dosaženo jednotným přístupem SSO. Pokud politiku přidělování zdrojů a ověřování identity uživatele předáme z uživatele na systém, pak se jedná o výrazné odlehčení požadavků na uživatele. Jediným požadavkem v tomto konceptu zůstává jedno (Single) přihlášení (Sign-on) do systému, které se provádí při přihlášení uživatele do intranetové sítě.

2.2 Architektura a autentizace

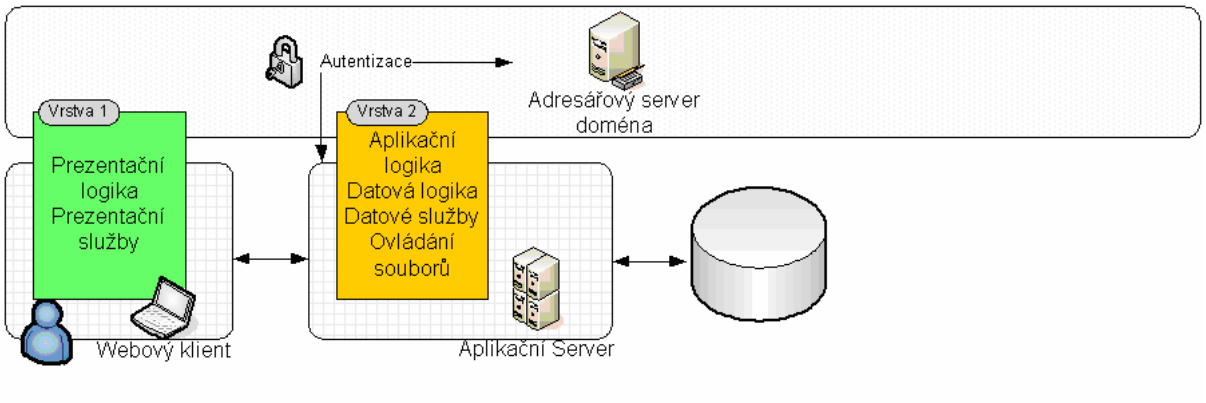
Způsob autentizace uživatele do webové aplikace se odvíjí od architektury aplikace. K ověřování dochází v takových místech aplikace a na takových vrstvách, aby byla komunikace klient-server bezpečná a dále aby ověření postihlo všechny vrstvy architektury aplikace, ke které má mít daný uživatel přístup. Pokud se tedy jedná o dvouvrstvou architekturu klient – server s aplikačním serverem, který přímo přistupuje pomocí aplikační logiky k databázovému serveru, obsahuje také tento aplikační server aplikační logiku k řešení autentizace uživatele v rámci aplikace. Pokud se ale jedná o třívrstvou architekturu, kde je logika dat a část aplikační logiky přenesena na Content Server (Databázový Server), je také na tomto serveru, aby řešil úkony spojené s ověřením uživatele.

Možnosti řešení Single Sign-on jsou tedy spojeny s architekturou služby, která je poskytována prostřednictvím webového klienta.

2.2.1 Typy architektur

Dvouvrstvá architektura klient – Server

Webová aplikace soustřeďuje veškerou logiku aplikace a správu dat na jeden aplikační server. Vzhledem k tomuto stavu je aplikační server vysoce zatížen. Schéma takového typu architektury je na obrázku.

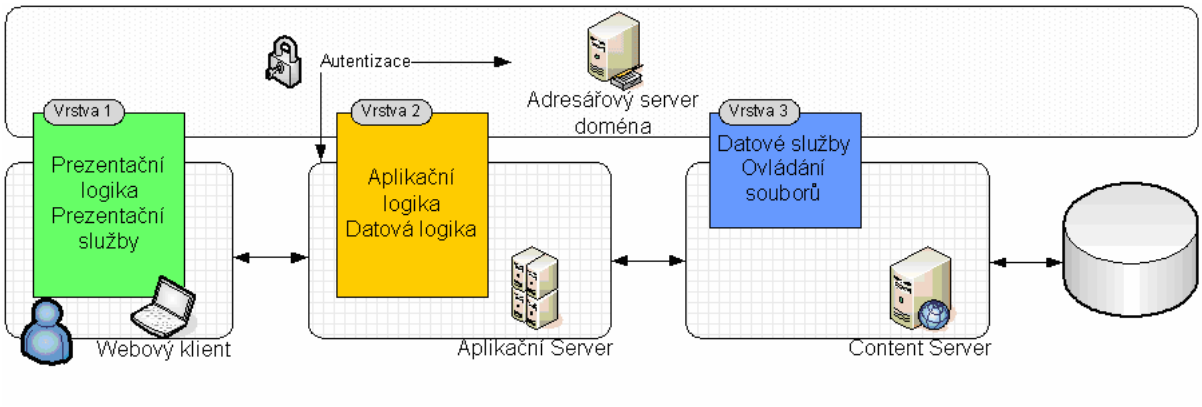


Obrázek 3: Dvouvrstvá architektura klient – server

Autentizace je soustředěna na aplikačním serveru. Aplikační server tvoří zároveň jedinou aplikační vrstvu aplikace a proto stačí, pokud dochází k ověření uživatele právě na tomto serveru pomocí aplikační logiky.

Třívrstvá architektura klient - server

Jak název architektury napovídá, obsahuje aplikace kromě klienta a aplikačního serveru i vrstvu, která poskytuje datové služby a ovládá soubory. Výhoda této architektury spočívá v celkové správě aplikace a rozdělení zátěže mezi aplikační a databázový server.

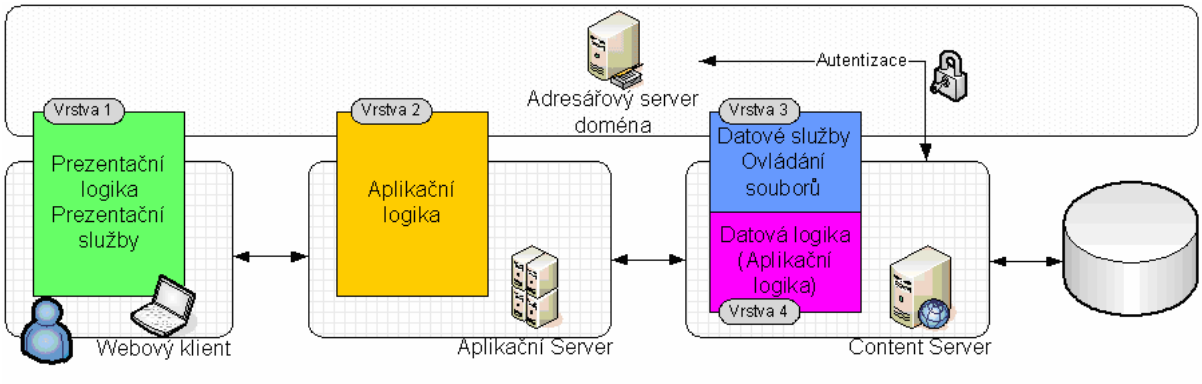


Obrázek 4: Třívrstvá architektura klient – server

Autentizace je řešena v rámci třívrstvé architektury podobně jako u modelu klient – server v rámci aplikační logiky na aplikačním serveru.

Vícevrstvá architektura klient – server

Architektura dělí oproti třívrstvé architektuře databázový server na Content Server (Content Management) a databázi.



Obrázek 5: Vícevrstvá architektura klient – server

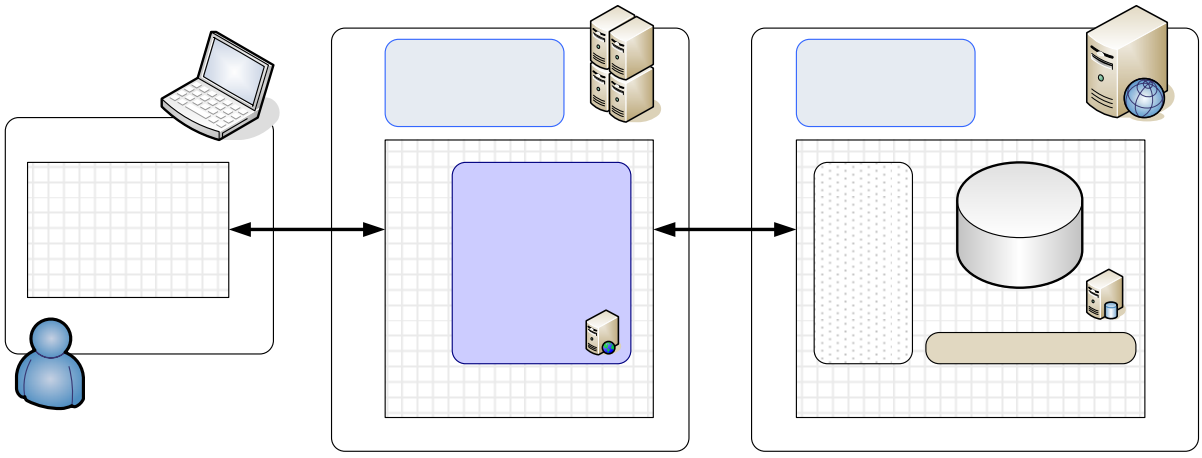
V rámci Content Serveru je soustředěno jak ovládání souborů a datových služeb, tak i datová logika. V některých případech obsahuje aplikační server pouze služby webového serveru (např. Tomcat a J2EE aplikace) a ostatní aplikační logika je přenesena také na Content Server.

Autentizace je v případě čtyřvrstvé architektury řešena na straně Content Serveru. Jedná se o nejsložitější řešení z pohledu přenosu utajovaných identifikačních údajů v rámci aplikace. Údaje po zadání uživatelem přejdou na aplikační server, kde je třeba tyto údaje z web requestu vyjmout a poslat je ke kontrole na Content Server. Teprve zde pak dochází k ověření těchto údajů a zaslání příslušné odpovědi zpět. Celá tato komunikace musí zabezpečená (např. SSL).

2.2.2 Architektura WEBACS

V rámci dalšího textu se budeme více zabývat jednou z možných architektur, která patří do vícevrstvého typu architektury – WEBACS (**Web A**plikační a **C**ontent **S**erver).

Architektura je základem J2EE aplikací, které jsou robusní svým zaměřením, rozsáhlé svými možnostmi a které důsledně oddělují aplikační server od datové logiky, datových služeb a metod pro práci s obsahem databáze.



Obrázek 6: Schéma architektury WEBACS

Webový klient – prohlížeč (browser), který komunikuje přes http protokol s webovým serverem. Vzhledem k povaze SSO protokolů bude má architektura klienta, který běží na operačním systému Windows. Prohlížeč musí splňovat podporu protokolů NTLM a Kerberos (např. Internet Explorer).

Aplikační Server – server postavený na J2EE platformě, jejímž základem z pohledu webové aplikace je servlet /JSP container. Příkladem takové konfigurace může být například Tomcat. Aplikační server může běžet na jakémkoliv operačním systému. Podpora SSO zde není závislá na systému, ale na nastavení a doinstalování modulu, který bude data směrovat na autetizaci na Content Server.

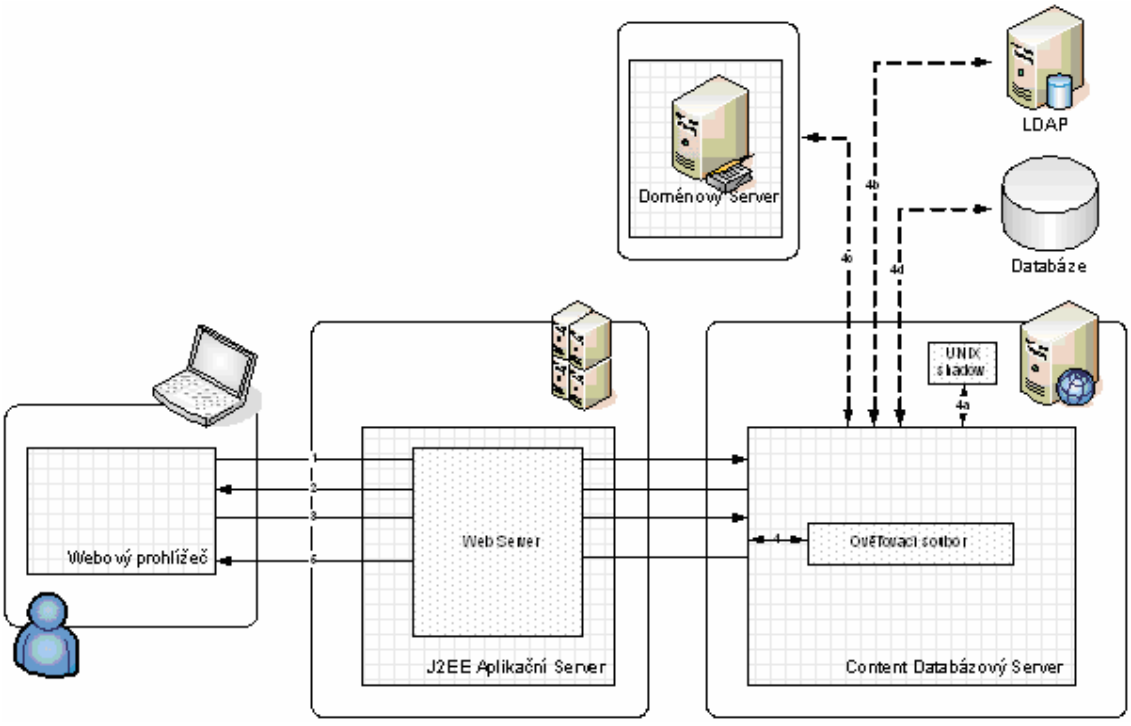
Content Server – server pro správu obsahu databáze. Součástí je mj. Ověřovací soubor (security plugin), který umožňuje různé způsoby úprfav.

Architektura WEABACS je součástí určité domény.

2.3 Řešení bezpečnosti autentizace

WEBACS architektura poskytuje Basic Auth autenetizační mechanismus, který využívá uživatelské jméno a heslo pro porovnání s údaji, které jsou uloženy na příslušném místě domény (Active Directory). Autentizace spočívá v několika krocích, do kterých jsou zapojeny všechny vrstvy architektury. Webový klient vznáší požadavek (request) na určitý zdroj. Na Aplikačním Serveru je tento požadavek transformován do takové podoby, aby jej bylo možné přenést na Content Server, na kterém dochází k vlastnímu ověření identity klienta. V tomto je základní rozdíl oproti třívrstvé architektuře, která soustřeďuje autentizační logiku na aplikačním serveru a databázový server slouží jako uložiště dat.

Následující diagram ukazuje komunikaci v rámci Basic Auth autentizace a možnosti ověření klienta. V rámci Content Serveru je v diagramu vidět Ověřovací soubor. Jedná se o modul, který řeší samotnou autentizaci a jehož výstupem je hodnota boolean, která značí, zda proběhla identifikace v pořádku. Tento soubor může mít u různých aplikací různou podobu a rozdílnou funkčnost. V našem případě bude soubor přijímat základní tři parametry, kterými jsou uživatelské jméno, uživatelské heslo a způsob autentizace.



Obrázek 7: Autentizace v architektuře WEBACS

Autntizace probíhá v následujících krocích:

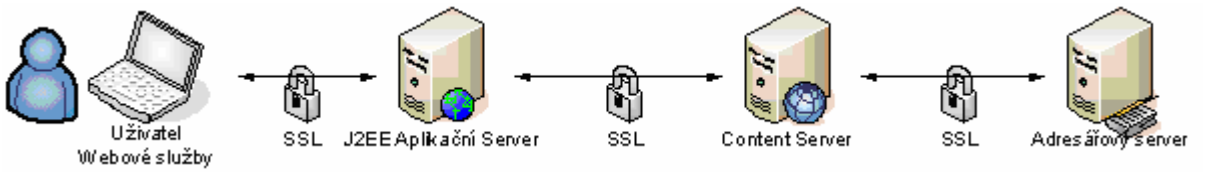
1. HTTP požadavek (request HTTP get) je přesměrován na Content Server
2. odpovědí je login formulář
3. uživatel naplní formulář autentizačními údaji a HTTP get je znovu poslán na Content Server
4. Content Server předá credentials Ověřovacímu souboru s parametrem, dle něhož soubor rozpozná, jakým způsobem má uživatele ověřit – inline (4a) v databázi Content Serveru, domain (4b) oproti doménovému serveru, unix (4d)
5. pokud je přístup k požadovanému zdroji povolen, zašle server zpět požadovaný obsah na prohlížeč a zároveň zasílá session cookie, která umožňuje opětovné ověření uživatele v rámci dané session

Během procesu ověřování uživatele je jméno a heslo posláno v rámci WEBACS architektury na Content Server, který obsahuje nástroje na autentizaci uživatele. Možností, jak uživatele ověřit, však může server poskytovat více. Definice těchto možností je obvykle uložena v databázové tabulce uživatelů, kde je u každého záznamu vybrán administrátorem způsob, jakým má být daný uživatel autentizován. Poté, co je způsob ověření vybrán, jsou autentizační údaje zaslány příslušné službě. Nejjednodušší případ takové služby je spouštěcí soubor (exe). S takovým souborem jsme se již setkali jako s Ověřovacím souborem. Může se jednat například o kód psaný v jazyku C, kterému se jako parametry předávají jméno a heslo uživatele spolu s příznakem definujícím způsob autentizace. Výstupem takového souboru je pak boolean hodnota, která vrací TRUE pokud je uživatel ověřen kladně, FALSE pokud je uživatel ověřen záporně. Dále předpokládejme, že soubor obsahuje funkce, které řeší všechny požadované možnosti autentizace. Struktura takového souboru pak bude vypadat následovně:

```
Boolean OverovaciSoubor
{
        boolean result = false;
        if (LDAP autentizace) {
                result = overeniProtiAdresarovemuServeru(uživatelJméno,uživatelHeslo,jménoServeru,
                ldap_port,bind_search_dn,atribut);
        }else  if (local autentizace)
                esult = overeniProtiLocalBazi(uživatelJméno,uživatelHeslo);
        }else if (domain autentizace) {
                result = overeniProtiDomene(uživatelJméno,uživatelHeslo,doména,pdc,bdc);
        }
        return result;
}
```

Autentizace je proces, který vede k potvrzení identity uživatele. Tento proces je v rámci intranetových sítí i mimo ně veden po síti. Vzhledem k této skutečnosti je třeba, aby komunikace mezi klientem a serverem byla patřičně zabezpečená a aby nebylo možné zjistit heslo nebo nějakou jinou důležitou informaci díky odposlechnutí komunikace na síti.

V praxi se používá zabezpečení pomocí kryptovacích algotitmů. Mezi nejpoužívanější je komunikace na bázi SSL (Secure Socket Layer).



Obrázek 8: SSL komunikace v architektuře WEBACS

SSL je protokol, resp. vrstva, vložená mezi vrstvu transportní (např. TCP-IP) a aplikační (např. http). Protokol spočívá ve vytvoření SSL spojení (session), nad kterým je komunikace mezi klientem a serverem zabezpečená – šifrovaná. Další informace v literatuře []

3 Řešení Single Sign-on

3.1 Definice SSO

Single Sign-on (SSO) je metoda řízení přístupu, která uživateli umožňuje jednou ověřit autentičnost a získat tak přístup ke zdrojům různých softwarových systémů [8].

Web Single Sign-on (Web-SSO) je metoda řízení přístupu, která pracuje výhradně s aplikacemi a zdroji, ke kterým se přistupuje prostřednictvím webového prohlížeče [9]. Přístup k webovému zdroji je odchycen buď použitím webového proxy serveru nebo instalací určité komponenty na každý z žádaných web serverů.

3.2 Základní principy SSO

Jednotné přihlášení – umožňuje uživateli využít pouze jedno přihlášení k přístupu k zabezpečeným aplikacím, které důvěřují přihlášení do domény.

Přehledná správa – administrátor intranetové sítě má možnost spravovat přístup k jednotlivým aplikacím domény z jednoho místa. Přehledná správa souvisí s centralizací bezpečnostní politiky na trust servery (např. KDC).

Omezení nechtěného přístupu – vzhledem k povaze SSO a snížením nároků na uživatele aplikací tím, že je uživatel povinen znát pouze jedno heslo, je zvýšena bezpečnost intranetové sítě

Volba autentizace – pokud není aktuálně možné využít možnost SSO, pak má řešení autentizace nabídnout uživateli Basic Auth model autentizace.

Využití systémových prostředků – většina implementací SSO využívá ve větší nebo menší míře systém pro získání a ověření uživatele (Windows credentials).

Úpravy aplikací – každá aplikace nebo služba, na kterou má být aplikována metoda SSO, musí být upravena. Buď se jedná o úpravu konfigurace aplikace, nebo se jedná o doinstalování celých modulů.

3.3 SSO v podnikové sféře

V podnikové sféře je v dnešní době široká poptávka po kvalitních řešeních Single Sign-on. Na scénu přicházejí velké firmy (jako CA), které řeší tuto problematiku robusním způsobem. Takové řešení však například bankovní instituce příliš neláká. Hlavním důvodem je zejména obava z případných velkých změn v rámci firemní sítě a její bezpečnosti. Proto i velké instituce poptávají taková řešení

Single Sign-on, která by byla jednoduše aplikovatelná a která by nenarušila jejich dosavadní koncept bezpečnosti.

Zejména ve finanční podnikové sféře převládá zároveň názor, že podnik raději zaplatí za vytvoření řešení na míru než kupovat hotová řešení, jejichž know-how není jen otázkou pracovníků daného podniku. Z těchto a dalších důvodů je v dnešní době silná poptávka po kvalitních, bezpečných a přitom jednoduchých řešeních Single Sign-on.

Stávající řešení ve firmách se zpravidla dělí na dva typy SSO. Prvním řešením je využití protokolu NTLM a druhým řešením je protokol Kerberos. Až na některé výjimky platí, že řešení na bázi NTLM způsobují uživatelům spíše problémy. Je to způsobeno charakteristikou aplikace protokolu (jak bude popsáno dále). To je svým způsobem pochopitelné, protože se jedná o jednodušší aplikaci než Kerberos. Z těchto a dalších důvodů je dnes výhradně poptávka po řešení na bázi protokolu Kerberos popřípadě větších centralizovaných řešení, jejichž politika se dá vestavět do již existujících sítí bez nějakých citelnějších změn v architektuře intranetové sítě.

3.4 Základní koncepty SSO

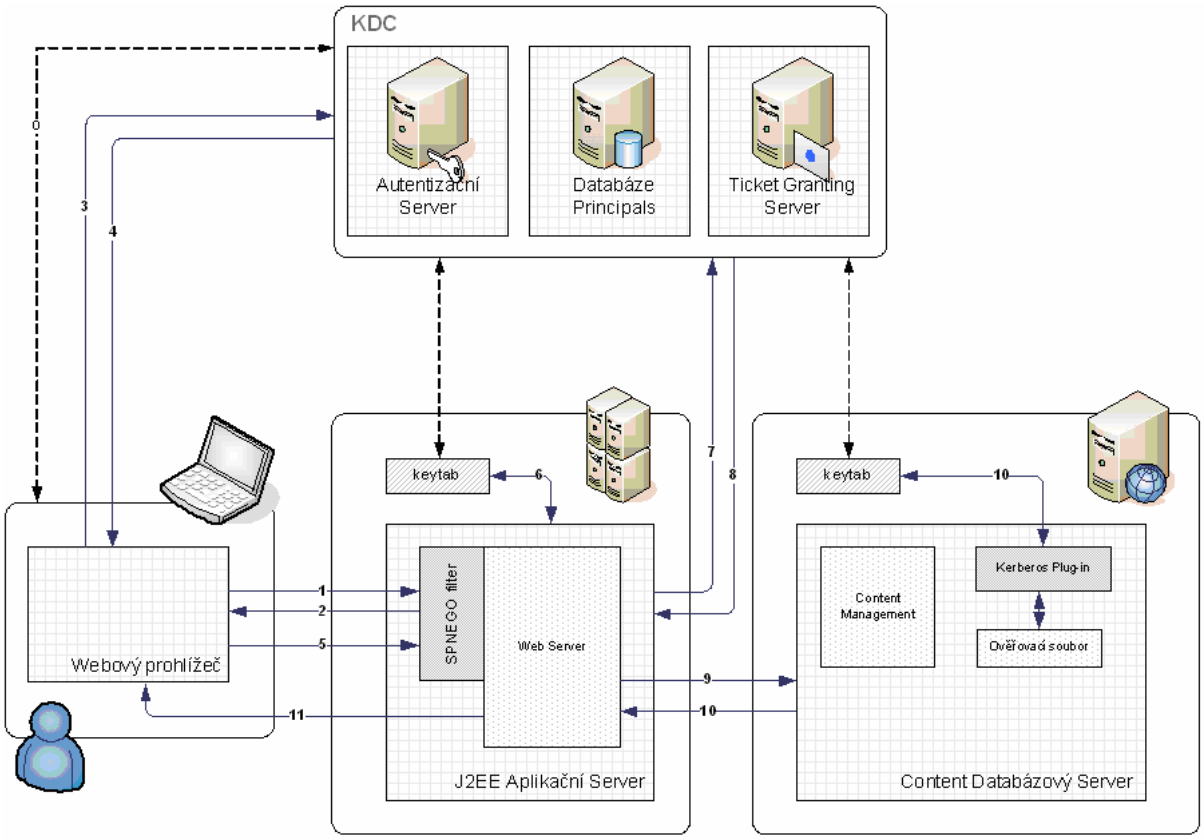
Podkapitola se zabývá základními koncepty, které lze najít v různých řešeních SSO pro webové aplikace. Typy konceptů jsou odvozeny od architektury jednotlivých implementací jednotného přihlašování. V rámci každého konceptu bude poukázáno na způsob procesu přihlašování do aplikace (až na případ NTLM se jedná o architekturu WEBACS), dále budou napsány výhody a nevýhody konceptu a na závěr každého konceptu bude ukázán příklad takového řešení.

Mezi základní koncepty patří tyto:

1. Kerberos SSO
2. Centralizované SSO
3. Systémové SSO
4. Enterprise SSO

3.4.1 Kerberos SSO

Koncept Kerberos (WinLogon) počítá s autentizací na úrovni Domain Controlleru. V rámci přihlášení uživatele k patřičné doméně obdrží desktop klient od Domain Controller Serveru autentizační lístek TGT (Ticket Granting Ticket). Tento lístek pak slouží pro přihlášení k dalším zdrojům a službám dané domény.



Obrázek 9: Kerberos SSO

Popis komunikace k diagramu:

0. Během přihlášení uživatele k PC, je vytvořen Kerberos TGT lístek a je uložen do Kerberos ticket cache. Lístek TGT je pak použitý pro komunikaci s KDC (Key Distribution Server).
1. Request bez autorizace na Webový Server. Request neobsahuje žádný Kerberos Service Ticket.
2. SPNEGO filter odchytí request a odpoví Unauthorized 401 se SPNEGO negotiation požadavkem.
3. Webový prohlížeč vznese požadavek na KDC s použitím TGT lístku a obdrží Kerberos Service Ticket pro poptávanou službu.
4. KDC pošle Kerberos service ticket zpět Webovému prohlížeči s použitím HTTP hlavičky.
5. Webový prohlížeč zašle Kerberos Service Ticket aplikačnímu serveru.
6. Webový server autentizuje uživatele na základě nakonfigurovaného souboru keytab.
7. Webový server zašle dotaz na KDC s požadavkem na Kerberos Service Ticket pro autentizaci na Content Serveru.
8. KDC pošle Kerberos Service Ticket zpět Webovému serveru.
9. Webový server zašle příslušný Kerberos Service Ticket na Content Server pro autentizaci.
10. Content Server autentizuje uživatele prostřednictvím Kerberos pluginu, který ověří shodnost se souborem keytab. Pokud je Service Token validní, povolí Content Server uživateli přístup ke zdroji.

11. Webový Server vrací prohlížeči požadovaný zdroj.

Výhody konceptu Kerberos SSO:

- Jedná se o čisté řešení SSO, které využívá autentizační údaje při logování k desktopu.
- Primární autentizační koncept v prostředí Windows 2000/2003.
- Je implementovatelný na všechny serverové operační platformy.
- Možnost využití v různorodých prostředích (více domén, různé platformy).
- Možnost delegované autentizace, při které se služba vydává za klienta k přístupu k další službě.
- Důvěra založená na držení bezpečnostních tokenů. Takové řešení pak může být integrováno s použitím SAML rozhraní do SOA, kde jsou autentizační a autorizační funkce dostupné jako webové služby k příslušným aplikacím.

Nevýhody konceptu:

- Nepodporuje SSO řešení pro tlustého klienta.
- Nepodporuje autorizaci. Řešení ale dokáže transportovat specifická data (např. PAC) generované a používané nějakou jinou autorizační službou.

Jako příklad lze uvést například Kerberos MIT, Heimdal, Domain Controller.

Příklad implementace:

Solfit je Švýcarská firma založená v roce 1997 a poskytující integraci, document management a identity management služby. Solfit se zaměřuje na řešení SSO pro Oracle, SAP nebo Documentum.

Jako příklad systému, který je založen zcela na využití Kerberos protokolu si můžeme uvést řešení Solfit. Řešení Solfit SSO je založeno na prostředí Active Directory a využívá vestavěné komponenty na Aplikační Server a na Content Server. Aplikační Server obsahuje Solfit SPNEGO filter, který je kooperován se security frameworkem sarveru. Na straně Content Serveru je na Ověřovací soubor navázán Solfit Plugin, který umí ověřit credentials oproti keytab souboru. Ostatní komunikace je stejná jako u Kerberos řešení SSO.

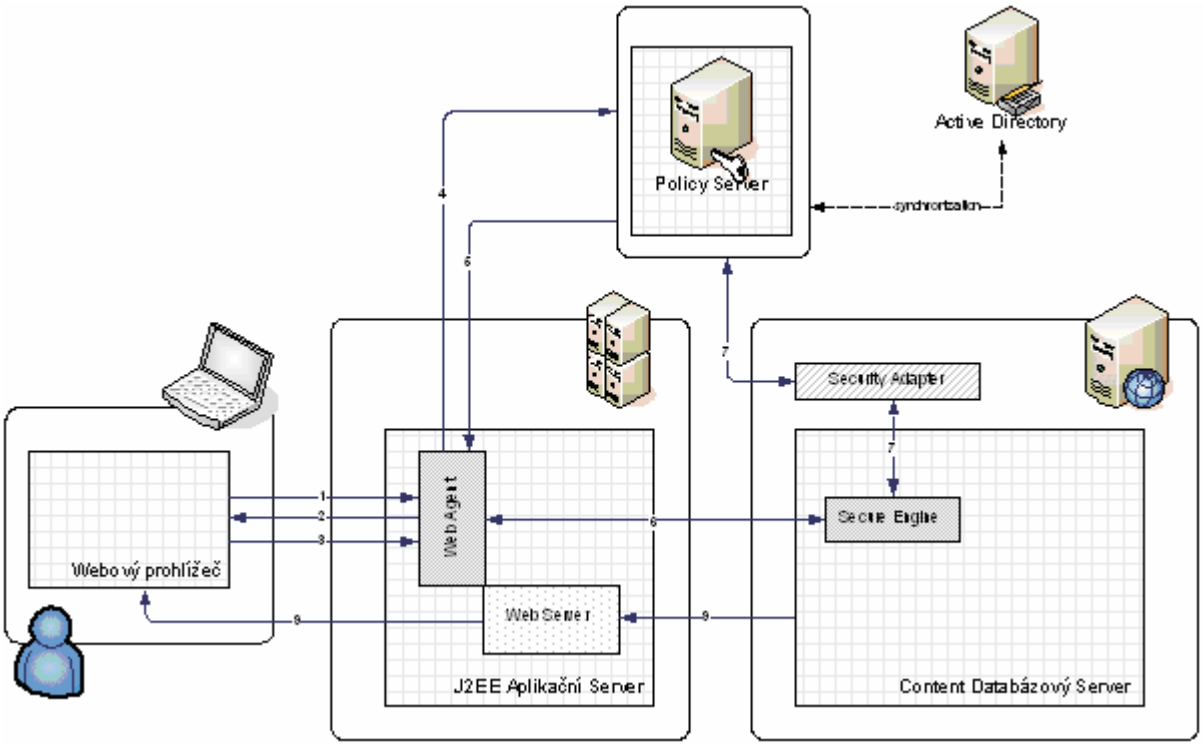
Solfit je typická Kerberos architektura, která je založena na komunikaci SPNEGO. Tu zajišťuje volání filteru Aplikačního Serveru. Po vykomunikování Kerberos lístku je tento lístek ověřen oproti keytab souboru a pokud proběhne ověření v pořádku, pak se Server dotáže KDC na lístek pro Content Server. Tento lístek je předán Aplikačnímu Serveru a poslán na Content Server. Content Server stejným způsobem ověří zaslaný lístek a pokud se řešení shodují, pak proběhla autentizace vpořádku.

Podpora implementace:

Web Application Server:	BEA WebLogic, Tomcat, WebSphere
Client	SPNEGO compatible Browser (IE, Firefox)
Client operating system	Windows 2000 SP4, XP SP2, 2003 SP1
Active Directory	2000, 2003
Support of multi domain	yes

3.4.2 Centralizované SSO

Koncept počítá s tím, že existuje jeden centrální Policy Server, který řeší autentizační požadavky od speciálních puginů – SSO agentů, kteří jsou instalováni na příslušných webových serverech. SSO agent zachytí neověřený web request a deleguje autentizaci a autorizaci proti centrálnímu Policy Serveru.



Obrázek 10: Centralizované SSO

Popis komunikace k diagramu:

1. Request je HTTP GET bez validní authentizační informace (smsession cookie a hodnota hlavičky uživatele).
2. Request klienta na chráněný Ewbový zdroj je řešen Webovým agentem, který pošle zpět modifikovaný login screen.

3. Uživatel vyplní jméno a heslo a vybere bázi, se kterou chce pracovat. Prohlížeč pošle tyto credentials jako součást HTTP hlavičky na Web Server.
4. Webový Agent zavolá eTrust Policy Server pro autentizaci a autorizaci požadavku.
5. eTrust Policy Server autentizuje uživatele s použitím uživatelského jména a hesla a vygeneruje autentizační lístek, který pošle jako session cookie v HTTP hlavičce zpět Webovému Agentovi.
6. Web Agent zformátuje credentials do SMSESSION tokenu a přesměruje jej na Content Server.
7. Content Server zavolá Secure Plugin pro autentizaci. Plugin pak volá Policy Server, který ověří SMSESSION token a tím ověří uživatele.
8. Jestliže je přístup na Content Server povolen, je poslán požadovaný obsah zpět prohlížeči. Autentizovaná session data jsou poslána v session cookie k dalšímu použití.

Výhody konceptu:

- Řešení je pro čistě webově založené aplikace.
- Agenti existují jen pro známé webové a aplikační servery.
- Možnost využití v různorodých prostředích (více domén, různé platformy).

Nevýhody konceptu:

- Potřeba zvláštního agenta na každou aplikaci, která umožňuje SSO.
- Typicky nepoužívá logovací údaje do desktopu.
- Nepodporuje SSO řešení pro tlustého klienta.

Příklad implementace - CA Netegrity SiteMinder

eTrust SiteMinder je access management software řešení, které poskytuje centralizované bezpečnostní služby pro uživatelskou autentizaci a přístup k webovým aplikacím. Základem řešení je centrální Netegrity SiteMinder Policy Server. Základním rysem řešení je podpora Single Sign-on, Strong Authentication Management, centralizované řešení autorizace a auditu a Identity federation.

Pokud se podíváme na jednotlivá řešení SSO na J2EE, pak by této implementaci SSO odpovídalo centralizované řešení SSO. Web Agent na straně Aplikačního Serveru je v rámci SiteMinder Netegrity Web Agent a Security Adapter je Netegrity plugin.

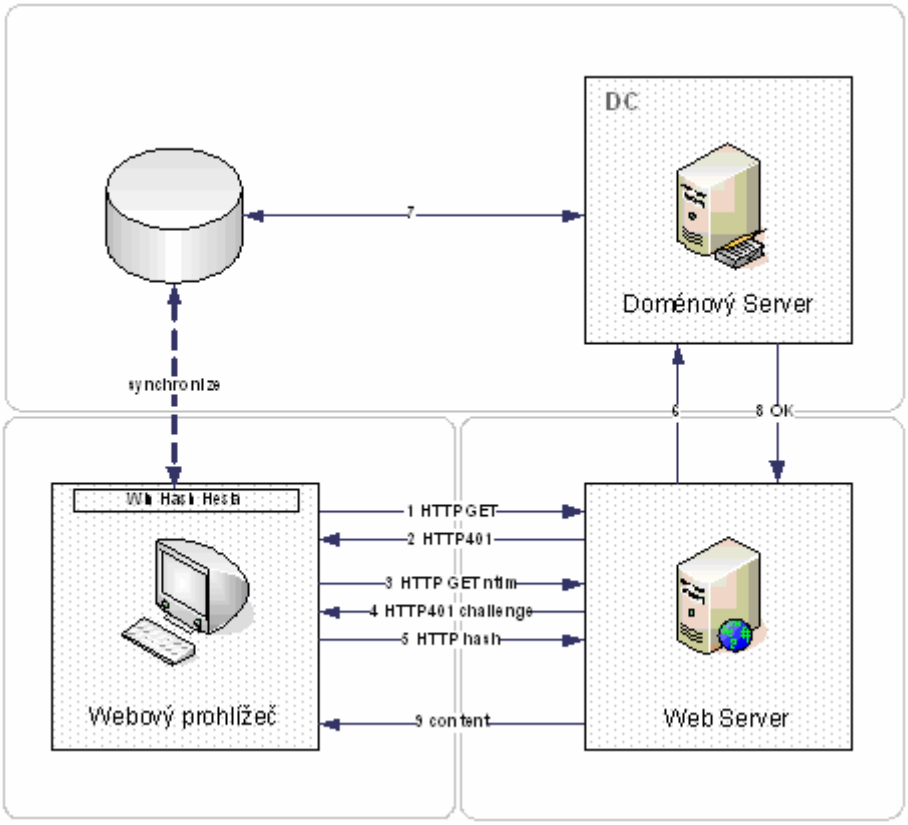
Podpora implementace:

Web Application Server:	BEA WebLogic, Tomcat, WebSphere
Client	IE, Firefox

Client operating system	Win NT, 2000, Solaris, HP-UX, Red Hat Linux, AIX
Active Directory	2000, 2003
Support of multi domain	yes

3.4.3 Systémové SSO (NTLM)

Koncept systémového řešení SSO je postaven na vlastnostech operačního systému klienta a někdy i serveru. Jako příklad lze uvést NTLM komunikaci. Pokud webový klient vrátí serveru na základě žádosti o autorizaci potvrzení o možnosti NTLM komunikace znamená to, že klientský browser umí komunikovat protokolem NTLM. Dále komunikace NTLM předpokládá vytažení hashe hesla uživatele z počítače s operačním systémem Windows. Jedná se tedy o implementaci SSO, která je silně závislá na systému.



Obrázek 11: Systémové SSO

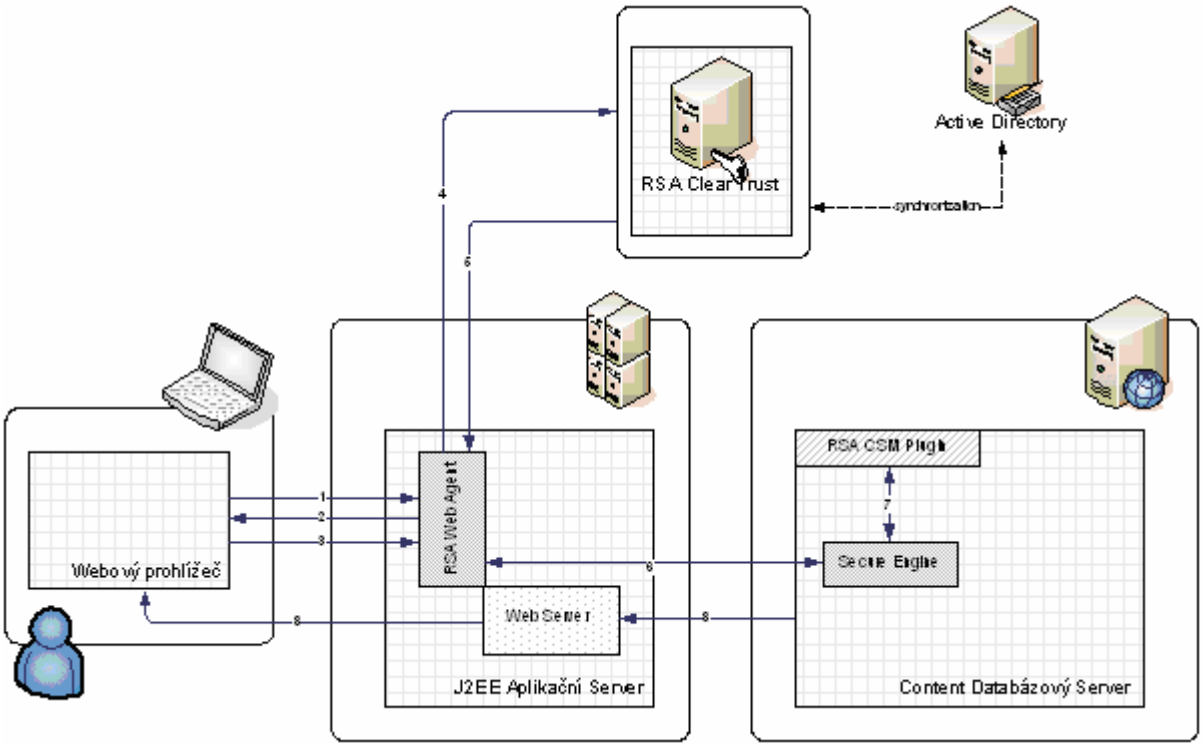
Popis komunikace k diagramu:

1. webový prohlížeč posílá HTTP GET požadavek na Web Server
2. Web Server zjišťuje že anonymní přístup nemůže použít a odesílá zpět HTTP 401 Access Denied. Zároveň nabízí klientovi mimo jiné možnost NTLM autentizace

3. klient si vyžádá stránku znovu a uvede jako způsob autentizace NTLM
4. Web Server odpovídá opět s HTTP 401 Access Denied a doplňuje náhodnou hodnotu pro klienta ("CHALLENGE")
5. klient zakóduje CHALLENGE zaslaný serverem pomocí hashe logon hesla uloženého na klientovi a vytvoří unikátní hash pro tuto transakci. Tento hash pošle serveru přes již otevřené spojení ("Keep alive"). Tím je zaručeno, že jedině uživatel se správným heslem a správnou náhodnou hodnotou mohl vytvořit tento hash
6. Web Server pošle hash, jméno uživatele a CHALLENGE na Domain Controller
7. Domain Controller zakóduje CHALLENGE pomocí hash hodnoty uživatelského hesla a pokud se výsledek rovná zaslané hash hodnotě ze strany Web Serveru, proběhla autentizace v pořádku
8. Domain Controller posílá výsledek autentizace na Web Server
9. pokud je výsledek autentizace kladný, zasílá Web Server klientovi požadovaný obsah

3.4.4 Enterprise SSO

Autorizační software RSA ClearTrust je jednotným řešením správy, které zajišťuje bezpečný přístup ke zdrojům založeným na webu. Autorizační software RSA ClearTrust centrálně kontroluje a řídí přístupová práva uživatelů ke zdrojům na webu [12]. Koncept Enterprise využívá desktop agenta, který odchytává login požadavky přístupu k určitému zdroji a sám je plní jménem a heslem.



Obrázek 12: Enterprise SSO

Popis komunikace k diagramu:

1. Request bez autentizace a platné RSA autentizační informace (session cookie).
2. Uživatelský požadavek na chráněný webový zdroj je odchycen RSA ClearTrust ASM Agentem, který zašle zpět modifikovaný login screen.
3. Uživatel vyplní jméno a heslo a vybere bázi, se kterou chce pracovat. Prohlížeč pošle tyto credentials jako součást HTTP hlavičky na Web Server
4. RSA ClearTrust ASM Agent se táže RSA ClearTrust Server na autentizaci a autorizaci požadavku.
5. RSA ClearTrust Server ověří uživatele s použitím uživatelského jména a hesla a vytvoří SSO session token, který pošle zpět jako session cookie v HTTP hlavičce na RSA ClearTrust.
6. RSA ClearTrust Agent vezme autentizační parametry a zašle je na Content Server.
7. Content Server zavolá RSA Plugin, který ověří SSO session token (pravděpodobně použitím sdíleného klíče bez potřeby volat RSA ClearTrust Server.
8. Jestliže je přístup na Content Server povolen, je poslán požadovaný obsah zpět prohlížeči.

Řešení pro architekturu WEBACS je postaveno na dvou oddělených modulech – ASM a CSM.

ASM – Application Server Module – běží jako část služeb pro webového klienta na straně Aplikačního Serveru.

CSM – Content Server Module – běží jako součást služeb na straně Content Serveru.

Výhody konceptu:

- Jedná se o čisté řešení SSO, které využívá autentizační údaje při logování k desktopu.
- Umožňuje integrovat téměř všechny aplikace v rámco řešení SSO včetně tlustých klientů.
- Možnost využití v různorodých prostředích (více domén, různé platformy).

Nevýhody konceptu:

- Je třeba instalovat a konfigurovat na každém stroji v síti zvlášť.
- Řešení se spoléhá na bezpečnost klienta. To může mít za následek zvýšení rizika napadnutí bezpečnosti intranetové sítě ze strany uživatelských desktopů.

Jako příklad lze uvést například RSA Sign-On Manager.

Podpora implementace:

Web Application Server:	BEA WebLogic, Tomcat, WebSphere

Client	MS Internet Explorer 6.0
Client operating system	Win NT, 2000, Solaris, Linux, HP-UX a IBM AIX
Active Directory	2000, 2003, Sun Java System Directory 5.2, Novell eDirectory 8.7.1 or 8.7.3
Support of multi domain	yes

4 Autentizační protokoly SSO

4.1 NTLM a Kerberos

Mezi základní autentizační protokoly splňující požadavky Single Sign-on patří NTLM a Kerberos protokol. Více o těchto protokolech bude napsáno v následujících kapitolách, zde budou vykresleny základní charakteristiky a rozdíly, které tyto protokoly obsahují.

V předchozí kapitole je popsáno několik základních řešení SSO – NTLM, Kerberos a Centralizované řešení. Vzhledem k tomu, že třetí řešení obvykle využívá protokolů NTLM a Kerberos pro aplikaci SSI, budeme se zabývat pouze těmito protokoly.

4.1.1 NTLM - základní charakteristika

Protokol NTLM je silně závislý na platformě (aplikaci), která chce tento protokol využívat. Z toho důvodu existuje poměrně velké množství různých uzpůsobení aplikací, které protokol využívají, a zároveň se setkáváme s různými řešeními ověřování klienta vůči PDC. Pokud máme například aplikační server Unix a na něm běžící Java Web Server, pak je třeba použít takové řešení pro ověření Windows klienta proti PDC, které nejen umí komunikovat protokolem NTLM, ale také ověřit vůči PDC. Pro tento případ je řešením například CIFS protokol. Pokud ale použijeme klasický IIS Server, máme řešení vestavěné v rámci homogenní sítě Microsoft.

Výhodou protokolu je jeho přehlednost a jednoduchost. To je způsobeno mimo jiné silnou aplikační závislostí na MS produktech. Jednoduchost i přehlednost spolu souvisejí. Protokol jako takový využívá vlastností Windows domény intranetové sítě. To má za následek možnost využívat základní funkčnost Windows systémů (MS operační systém klienta a Active Directory) na základní úrovni bez nutnosti řešení diference operačních prostředí systémů. Tímto odpadá jedna aplikační vrstva systému provozujícímu SSO. Tato výhoda je však nevýhodou v multioperačním prostředí nemluvě o nucené závislosti na MS Active Directory a Domain Controller.

Jelikož se zde zabýváme řešením na bázi J2EE Aplikačních Serverů, je třeba pohlížet na NTLM přes implementaci další vrstvy, která řeší komunikaci s klientem a doménou (např. knihovna jCIFS viz. dále). Komunikace mezi Aplikačním Serverem a klientem je udržována přes HTTP session. Problém ale nastává v komunikaci s doménovým serverem. Zde je třeba vytvořit session přes SMB protokol a kterou je třeba během jedné NTLM komunikace a ověření stále udržovat z důvodu získaného challenge. Tento stav není problémem, pokud se autentizuje Aplikační Server přímo vůči doméně. V případě WEBACS architektury ale vzniká problém udržení session. Ten lze vyřešit pouze tím, že se vytvoří Server SMB, který bude stále udržovat spojení s doménovým serverem zatímco si Aplikační Server bude žádat o vytvoření session, o získání challenge a po přeposlání NTLM údajů na Content Server zároveň ověření klienta. Z tohoto důvodu je NTLM autentizace možné použít pouze u

jednoduchých aplikací popřípadě složitějších s přidaným vlastním serverem, na kterém bude služba SMB běžet.

4.1.2 Kerberos - základní charakteristika

Kerberos je významný protokol díky své použitelnosti a nezávislosti na platformě. Této základní vlastnosti je dosaženo systémem serverů, které poskytují služby na bázi distribuce lístků a jejich ověřování. Každá služba nebo Aplikační Server, který chce využívat Kerberos pro SSO přístup, musí implementovat pouze soubor keytab a musí umožnit přetypování autentizace na použití Kerbera. Tato vlastnost je důležitá především u složitějších systémů, které mají více služeb a u kterých ověřování klienta provádí Content Server, jako je tomu například u architektury WEBACS.

Kerberos je zároveň velmi kompaktním protokolem, který staví na vrstvě KDC Serveru. Systém lístků (Kerberos ticket) je využit každou aplikací, která využívá Kerberos, a proto je možná komunikace i napříč operačními systémy. Pro jednoho klienta daného operačního systému stačí jednoduchá funkcionalita, která umí obdržet daný lístek, dekódovat jej a porovnat s obsahem keytab souboru. Pokud se jedná o delegovanou autentizaci, musí se umět daný klient dotázat KDC na lístek pro požadovanou další službu. A to je zároveň případ architektury WEBACS.

Implementace protokolu Kerberos jsou lehce dostupné pro základní operační systémy. Pro Unix jsou klíčové produkty MIT a Heimdal Kerberos, pro Windows doménu zase Windows Server 2000/2003implementace Kerbera. Co se prostředí Windows týče, je Kerberos od Windows Server 2000 primárním autentizačním protokolem (vystřídal dosavadní NTLM).

4.2 Základní porovnání autentizačních protokolů

protokol	**Kerberos**		**NTLM**	
Závislost na platformě	Unix - MIT, Heimdall, Windows - Windows Server 2000/2003	+	Windows doména, Windows klient OS	-
Přehlednost a jednoduchost	Využívá vrstvu mezi Domain Controller a Aplikačním Serverem	-	přímé využití prostředí	+
Použití na složitější aplikace	Možnost jakýchkoliv úprav	+	Pouze jednoduché aplikace	-
Implementace na Webový Server	Je nutné použít knihovnu podporující Kerberos a keytab	+	Je nutné použít knihovnu podporující NTLM	+
Implementace na službu	Ano	+	Ne	-
Koncepce protokolu	Centralizovaná, přehledná správa	+	Jedno-aplikační, malé možnosti rozšíření	-
Budoucnost protokolu	Ověřený koncept - jednoduchý a přehledný, primární ověřování MS Windows	+	NTLM2, omezené možnosti	-

5 NTLM

5.1 Základy protokolu NTLM

5.1.1 Mechanismus NTLM

NTLM je protokol pro ověřování uživatelů použitý v různých implementacích Microsoft síťového protokolu s podporou NTLM Security Suport Provider („NTLMSSP“). V Microsoft sítích je též používán jako integrovaný single sign-on mechanismus. NTLM používá SMB protokol (CIFS).

NTLM používá challenge-response mechanismus pro autentikaci, skrze který je umožněno klientovi ověřit jeho identitu bez zaslání hesla na server. Celý postup se skládá ze tří zaslaných správ, běžně označovaných jako Type 1 message (negotiation), Type 2 message (challenge) and Type 3 message (authentication). Jednoduše pracuje protokol následovně:

- Klient zasílá Type 1 message serveru. Zpráva obsahuje seznam protokolů a rysů podporovaných klientem a žádajících po serveru.
- Server odpoví zprávou Type 2 message. Ta obsahuje seznam dohodnutých rysů a protokolů, na jejichž bázi proběhne autentizace. Nejdůležitější složkou zprávy je challenge generovaný serverem.
- Klient odpovídá na challenge zprávou Type 3 message. Ta obsahuje několik informací o klientu, včetně domény, username. Dále obsahuje hash vzniklý pomocí challenge, kterým dojde k verifikaci klienta na straně serveru

5.1.2 NTLM Komunikace

NTLM komunikace se skládá ze zpráv, které obsahují na základě fyzické vrstvy hlavičku NTLM. Každá NTLM hlavička začíná řetězcem NTLMSSP. Poté následuje obsah zprávy, který můžeme začlenit do jedné ze tří typů zpráv (1, 2 nebo 3).

NTLM http komunikace probíhá následovně (příklad z praxe):

1: Client --> Server
GET /index.html HTTP/1.1

2: Client <-- Server
HTTP/1.1 401 Unauthorized
WWW-Authenticate: NTLM
Connection: close

3: Client --> Server NTLM <**base64-encoded type-1-message**>
GET /index.html HTTP/1.1
Authorization: NTLM XXX

4: Client <-- Server NTLM <**base64-encoded type-2-message**>
HTTP/1.1 401 Unauthorized
WWW-Authenticate: NTLM YYY

5: Client --> Server NTLM <**base64-encoded type-3-message**>
GET /index.html HTTP/1.1
Authorization: NTLM ZZZ

6: Client <-- Server 200 Ok

5.1.3 Zpráva typu 1

První zpráva obsahuje host name a jméno NT domény klienta. Struktura zprávy je následující.

byte	**0**	**1**	**2**	**3**
0	"N"	"T"	"L"	"M"
4	"S"	"S"	"P"	0
8	1	0	0	0
12	0x03	0xb2	0	0
16	domain length		domain length	
20	domain offset		0	0
24	host length		host length	
28	host offset		0	0
32	host string, domain string			

Host name a domain name jsou ASCII řetězce s velkými písmeny.

5.1.4 Zpráva typu 2

Je zpráva zasílaná serverem klientovi a její hlavní částí je challenge. Tento řetězec je pak použit pro vytvoření LM nebo NT response. Jedná se o 8mi bytové pole.

byte	**0**	**1**	**2**	**3**
0	"N"	"T"	"L"	"M"
4	"S"	"S"	"P"	0
8	2	0	0	0
12	0	0	0	0
16	message len		0	0
20	0x01	0x82	0	0
24	server challenge			
28				
32	0	0	0	0
36	0	0	0	0

Windows NT Challenge/Response je jednou z bezpečných cest jak určit, kdo vlastně vznáší požadavek na autentizaci. Windows NT Challenge/Response totiž neposílá heslo celou cestou skrz Internet (kde si ho může odposlechnout kdokoliv), hesla jsou před odesíláním zakódována algoritmem neumožňujícím reverzní postup. Jedná se o takzvané jednosměrné algoritmy. Výsledkem je tzv. hash z hesla - NT používá MD4 hashing algoritmus produkující 16ti bajtové (128bitů) hash kódy. Je teoreticky nemožné vzít hash a algoritmus a poté reverzně dojít k původnímu heslu. Hesla v tomto ohledu slouží jako privátní klíč. Pomocí hesla (hashe hesla uloženého na localu) a challenge dojde k vygenerování hash posloupnosti bytů.

IIS se pokusí použít Challenge/Response ověření pokud daný prohlížeč podporuje Challenge/Response komunikaci (např. IE). Selže-li tento pokus, pošle HTTP 401 Access Denied zprávu doplněnou přehledem možností autentikace - Challenge/Response nebo Basic. Internet Explorer si vždy vybere Challenge/Response oproti Basic. Ostatní prohlížeče naopak vždy Basic (protože nic jiného neumí).

5.1.5 Zpráva typu 3

Třetí zpráva obsahuje uživatelské jméno, host name, NT domain name a LM/NT responses.

byte	**0**	**1**	**2**	**3**
0	"N"	"T"	"L"	"M"
4	"S"	"S"	"P"	0

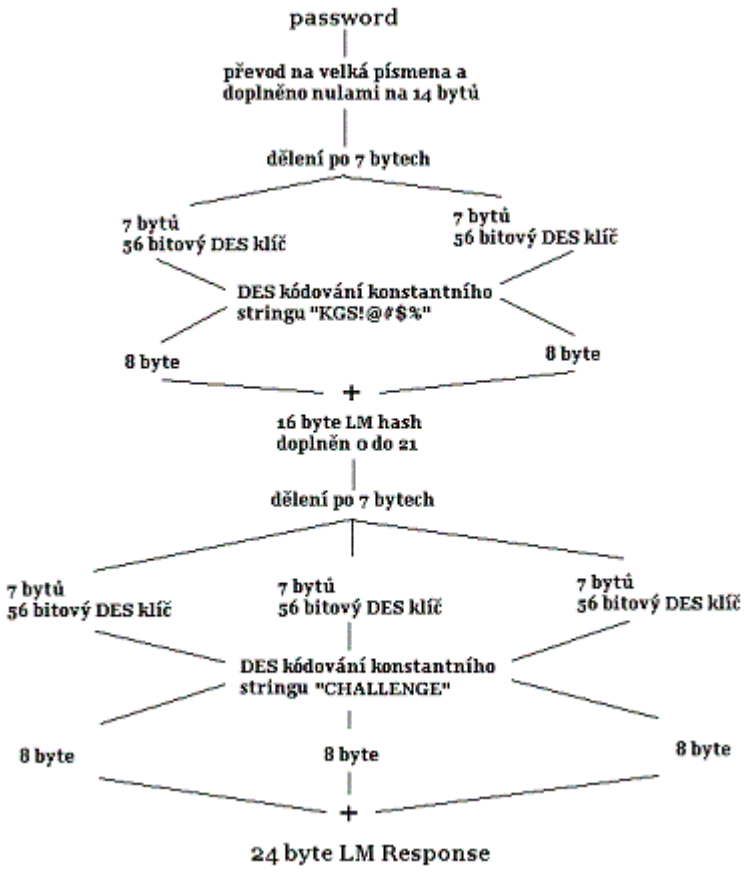
8	3	0	0	0
12	LM len		LM len	
16	LM offset		0	0
20	NT len		NT len	
24	NT offset		0	0
28	domain length		domain length	
32	domain offset		0	0
36	user length		user length	
40	user offset		0	0
44	host length		host length	
48	host offset		0	0
52	0	0	0	0
56	message len		0	0
60	0x01	0x82	0	0
64	domain string, user string, host string, LM response, NT response			

Host name, doména, user name jsou v Unicode kódování. Délka response řetězců je 24.

LM Response

LM hash (Lan Manager hash) je jeden z formátů, který používá Microsoft LAN Manager a Microsoft Windows k ukládání uživatelských hesel , které jsou kratší jak 15 znaků.

Ačkoliv je tento algoritmus založený na DES, může být v dnešní době již poměrně snadno prolomený díky dvěma mezerám v implementaci. V první řadě hesla, která jsou delší jak 7 znaků jsou děleny na dvě části a každá z těchto částí je heshovaná odděleně. Druhá nevýhoda spočívá v prvním kroku vytváření hashe – totiž v převodu znaků na upper-case. První případ umožňuje útočníkovi odděleně prolomit oddělené části hashe, druhý také značně zmenšuje množství možností na jeho prolomení.



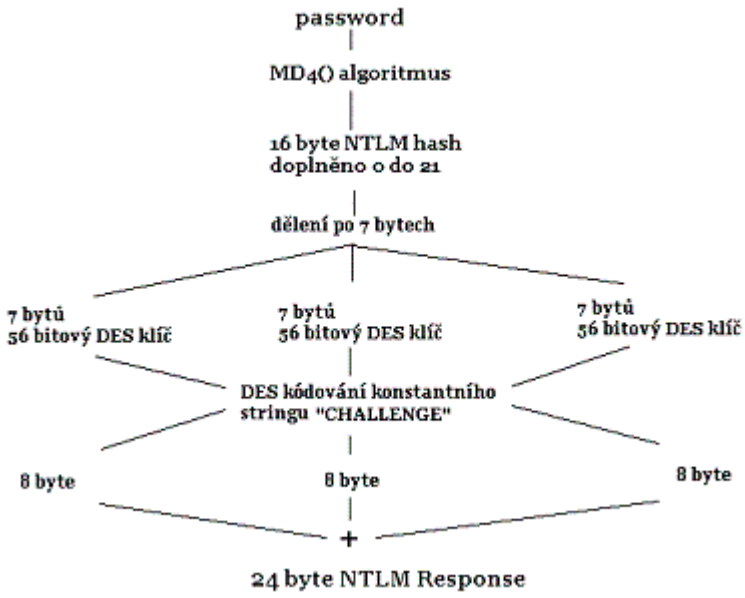
Obrázek 13: Proces vzniku LM hash

V dnešní době je snaha LM hash pokud je to možné nepoužívat. V operačním systému Windows Vista je použití LM hashe zcela eliminováno.

NT Response

Rozdíl od LM hash spočívá především v odstranění jeho nedostatků.

Co se týče dělení hashe na dvě části, je tento problém eliminován především použitím MD4 algoritmu, který pracuje nad heslem jako celkem. Druhou výhodou NT algoritmu je pak jeho práce s řetězcem, u kterého se nerozlišuje velikost písmen. Vstupem je tedy řetězec case-sensitive.



Obrázek 14: Proces vzniku NT hash

Bezpečnost NT response je podstatně větší jako u předchozího LM hashe. V dnešní době je ale i tento mechanismus vcelku zastaralý. Na dnešní poměry je z hlediska bezpečnosti vhodné použít NTLM2.

5.2 Autentizace NTLM

NTLM autentizaci můžeme rozdělit na dvě fáze. První fází je zasílání zpráv mezi MS browserem a Webovým Serverem a v druhé fázi se pak jedná o samotné ověření uživatele ze strany Domain Controlleru. Ověření je možné implementovat na základě SMB protokolu.

5.2.1 Proces NTLM autentizace

NTLM komunikace se účastní tři strany. Klient, Web Server a Doménový Server. Klient a Doménový Server patří do rodiny produktů Microsoft. Webový Server pak musí obsahovat podporu pro NTLM protokol a dále podporu pro komunikace s PDC. Komunikace s PDC je možno provést na základě SMB protokolu, který je vysvětlen v další kapitole.

NTLM autentizace je mechanismus, při kterém je dovoleno klientovi přistoupit k požadovanému zdroji s použitím Windows credentials a je typicky použitý v rámci intranetové sítě za účelem Single Sign-on. Historicky byl NTLM protokol podporován pouze klientem Internet Explorer, podpora NTLM však byla postupně přidána i do dalších Webových klientů.

Microsoft Internet Explorer má schopnost přenést NTLM hashované heslo přes HTTP session s použitím base 64 kódovaných NTLMSSP zpráv. Tento rys se týká MS IIS, ale Java Aplikační Server může díky CIFS také ověřit MSIE klienta proti Domain Controlleru.

Na schématu z kapitoly 3.5.1 je vidět proces autentizace využívající standardní IIS řešení Aplikačního Serveru. Komunikace mezi Webovým Serverem a Domain Controllerem však v případě řešení v jazyce Java bude obsahovat určitou funkcionalitu navíc.

5.2.2 CIFS/SMB protokol

Úvod do protokolu

Autentizace pomocí NTLM přístupu je založena na využití protokolů CIFS a SMB. SMB je protokol (Server Message Block) pro sdílení souborů, tiskáren a dalších komunikačních abstraktů mezi počítači. Protokol byl vyvinut firmou IBM v roce 1984, v roce 1986 však vývoj převzala firma Microsoft. V rámci produktů firmy Microsoft se stal SMB protokol základem pro sdílení tiskáren a souborů takových systémů jako byl DOS i Windows. Základy protokolu dalo IBM, tzv. SMB Core Protocol [11], který pak byl postupně firmou Microsoft vyvíjen až do dnešní podoby, kterou je protokol CIFS (Common Internet File System). Oba protokoly jsou navzájem disjunktní, tj. CIFS je nadmnožinou protokolu SMB. S protokolem CIFS také souvisí jeho open-source implementace, která je známá pod názvem Samba (Australian National University 1991). Jedná se o významné řešení z hlediska propojení Windows s prostředím UNIX (autentizace stanic Windows oproti Unix serveru). Vedle Samby vzniklo na základě otevřené specifikace CIFS (organizace SNIA) ještě několik dalších projektů implementujících protokol SMB. Mezi nejzajímavější projekt z hlediska využití pro autentizaci se stal jCIFS implementovaný v jazyce Java.

Bezpečnost

Základem pro autentizaci uživatelů prostřednictvím SMB protokolu je skutečnost, že před přístupem ke každému sdílenému síťovému prostředku na síti se musí uživatel serveru ověřit heslem. Výsledkem této autentizace je buď odmítnutí přístupu, nebo přijetí 16ti bitového identifikátoru uživatele pro další komunikaci.

V rámci autentizace mohou být hesla posílaná k serveru buď jako textový řetězec, nebo může výměna informací probíhat kryptovanou formou. První možnost se vyskytuje spíše u starších klientů a serverů. Ověřování hesla v rámci kryptované výměny hesla je založena na principu challenge-response. Princip komunikace je založen na zaslání náhodně vygenerovaného čísla klientovi, který použije heslo ke kryptovací operaci a výsledek pak odešle zpět na server. Vzhledem k tomu, že server zná identifikační údaje klienta, dokáže stejným postupem dojít k řetězci, který byl zaslán klientem. Porovnáním těchto řetězců se pak ověří pravost uživatele. Náhodné číslo, které je během vyjednávání generováno, se nazývá challenge. Klient vypočítá z uživatelského hesla 168 bitový session key, kterým challenge pomocí DES algoritmu zakryptuje, čímž vzniká 192 bitová odpověď, která je odeslána serveru. Session key může být z uživatelského hesla převedeného n velká písmena počítán jedním z následujících způsobů. Buď je možné vypočítat pomocí DES algoritmu jako tzv. LM

Session Key, nebo pomocí MD4 algoritmu jako NT Session Key. První způsob je použit v protokolu SMB a druhý v protokolu CIFS.

V rámci odpovědi na žádost o autentizaci je obdržena jednobitová informace, která značí, zda je uživatelské jméno a heslo platné.U některých případů se během autentizace získají ještě další data. Těmi mohou být např. jméno, popis uživatele, profil a další údaje. Tyto údaje lze získat v doménách Windows NT, které obsahují řadič domény. Tento server koordinuje práci serverů a obsahuje databázi účtů SAM (Security Account Manager). Během startu počítače se každý CIFS server v doméně přihlásí k PDC (Primary Domain Controller), který jej mechanismem challenge-response vyzve k autentizaci. Následuje tzv. NetLogon autentizace.

NetLogon autentizace

Klient kontaktuje CIFS server, který odešle klientovi nazpět challenge. Následuje vytváření response na straně klienta, které obsahuje zašifrování challenge pomocí uživatelova hesla a tuto čifru pak spolu s jménem uživatele vrátí zpět serveru. CIFS server však nezná uživatelovo heslo, protože je uloženo v SAM na PDC. Z toho vyplývá, že nemůže ověřit, zda je response správná. Proto pošle po zabezpečeném kanále response na PDC, který uživatele autentizuje.

Hrozby

Možnost u starších verzí protokolu SMB zaslat heslo v jednoduché podobě jako textový řetězec po síti nebudeme brát v potaz. To je z bezpečnostního hlediska nepřípustné. Zaměříme se tedy na kryptovanou formu komunikace.

Většina útoků na protokol SMB je rovnocená prolomení použitého kryptografického algoritmu. Útočník může podvrhnout vlastní challenge serveru a z jeho odpovědi se může pokusit zlomit funkci, která je jednorozměrná. V tomto mohou být slabinou protokoly MD4, MD5. Další hrozbou mohou být z pohledu systému příliš jednoduchá uživatelská hesla. Dále není protokol odolný vůči odposlechnutí přenášené informace, protože komunikace není šifrovaná. Pokud by bylo potřeba zaručit i šifrování komunikace klient – server, pak by bylo nutné vložit mezi transportní vrstvu a SMB protokol vrstvu, která by data šifrovala. K šifrování dat je možné použít např. SSL. Tato možnost však závisí na podpoře klienta (Samba, Windows NT)..V opačném případě je třeba komunikaci provádět v zabezpečeném síťovém prostředí a tuto síť oddělit od ostatního prostředí pomocí SSL proxy serveru [10].

5.2.3 Java NTLM autentizace

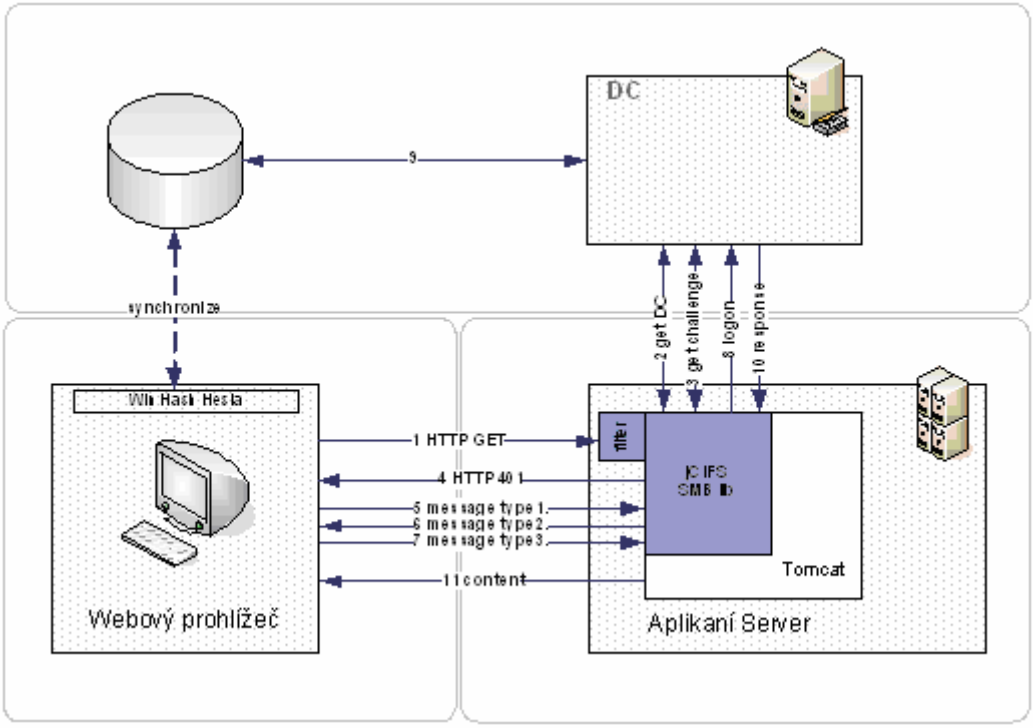
Pokud je použit Java Aplikační Server, je možné využít SMB protokolu k tomu, abychom byli schopni ověřit klienta. Schéma zůstává podobné, je ale třeba přidat funkcionalitu na Java Aplikační Server jak pro komunikaci s klientem (tedy podpora protokolu NTLM) a dále funkcionalitu pro

komunikaci s Domain Controllerem (podpora protokolu CIFS). Více o Java NTLM autentizaci v následující kapitole.

5.3 Implementace NTLM

Pro implementaci NTLM autentizace budeme používat knihovnu jCIFS projektu Samba. jCIFS je open source klientská knihovna, která implementuje CIFS/SMB síťový protokol v jazyce Java. CIFS je standardní protokol na platformě Wondows.

Hash hesla, který je generován při logování k pracovní stanici, která je součástí domény, bude přenesen díky request session vykomunikované knihovnou jCIFS a dále bude validován oproti PDC. Pro instalaci je pak třeba nastavit jCIFS Filter, který přesměruje ověření na knihovnu jCIFS na Aplikačním Serveru.



Obrázek 15: NTLM autentizace

5.3.1 jCIFS

Knihovnu jCIFS lze použít tak jak je v podobě archiv souboru jar. Po přidání knihovny na server a nastavením konfiguračních souborů web.xml a filtru může být knihovna použita pro autentizaci tak jak je. Dále je možné naprogramovat si vlastní podporu NTLM protokolu a SMB ověření na základě knihovny jCIFS.

Použití knihovny je ukázáno na servletu NtlmServlet, který implementuje knihovnu jCIFS. Servlet získá komunikací s klientem všechny zprávy NTLM komunikace.

```
import javax.servlet.*;
import jcifs.util.*;
...
public class NtlmServlet extends HttpServlet {
…
}
```

Na začátku servletu si definujeme challenge, spolu s náhodným řetězcem „nonce“, který bude pomocí hashe svého logon hesla hashovat. Devátý byte má hodnotu 2. Tato hodnota definuje, že se jedná o druhou zprávu zasálanou klientovi v rámci NTLM komunikace.

```
final private static byte[] CHALLENGE_MESSAGE =
    {(byte)'N', (byte)'T', (byte)'L', (byte)'M', (byte)'S', (byte)'S', (byte)'P', 0,
    2, 0, 0, 0, 0, 0, 0, 0,
    40, 0, 0, 0, 1, (byte)130, 0, 0,
    0, 2, 2, 2, 0, 0, 0, 0, // nonce
    0, 0, 0, 0, 0, 0, 0, 0};
```

Metoda processRequest zpracovává požadavek ze strany klienta. Z požadavku si bere hlavičku Authorization a zjišťuje, zda hlavička obsahuje nějakou možnost autentizace. Pokud neobsahuje, pak se nastaví status response na 401 a zároveň se hlavička nastaví na NTLM. Pokud hlavička obsahuje jinou možnost autentizace jak NTLM, pak dojde k ukončení.

```
protected void processRequest(HttpServletRequest req, HttpServletResponse res) {
        …
        String auth = req.getHeader("Authorization");

        if (auth == null) {
            res.setContentLength(0);
            res.setStatus(res.SC_UNAUTHORIZED);
            res.setHeader("WWW-Authenticate", "NTLM");
            res.flushBuffer();
            return;
        }

        if (!auth.startsWith("NTLM ")) {
            return;
        }
        …
}
```

Každá příchozí zpráva se nejdříve dekóduje pomocí base64.

```
byte[] msg = new sun.misc.BASE64Decoder().decodeBuffer(auth.substring(5));
```

Devátý byte definuje typ negotiation zprávy. Pokud se jedná o příchozí první zprávu, pak se posílá zpět klientovi zpráva typu dvě, která se skládá z typu autentizace a zakódovanou challenge strukturou. Pokud se jedná o třetí zprávu, dosáhli jsme požadovaného cíle a servlet se zobrazí bez chyby klientovi.

```
if (msg[8] == 1) {
        res.setContentLength(2);
        res.setStatus(res.SC_UNAUTHORIZED);
        res.setHeader("WWW-Authenticate",
        "NTLM " + new sun.misc.BASE64Encoder().encodeBuffer(CHALLENGE_MESSAGE));
        res.flushBuffer();
        return;
}

if (msg[8] == 3) {
}
```

Použitý kód používá napevno zvolený challenge s náhodným řetězcem nonce. V praxi však tento krok provádí SMB klient, který generuje různé řetězce challenge. V následujícím pseudo-kódu budou ukázány základy procesu NTLM autentizace s ověřením proti Domain Controlleru.

1. Vytvoření session na Domain Controller. Jako základní parametr se udává ip adresa Serveru (parametr domainController).

```
dc = UniAddress.getByName( domainController, true );
```

2. Získání challenge od Domain Controlleru. Funkčnost poskytuje třída SmbSession s metodou getChallenge(), které se předá jako parametr identifikátor získané session Domain Controlleru.

```
byte[] challenge = SmbSession.getChallenge( dc );
```

3. Získání NTLM struktury na základě zaslání tří zpráv komunikace mezi klientem a Aplikačním Serverem. Parametry metody authenticate jsou request a response odchycené filtrem a challenge z předchozího kroku. Výsledkem je struktura, která obsahuje mimo jiné LM a NT hash pro identifikaci klienta.

```
ntlm = NtlmSsp.authenticate( req, resp, challenge );
```

4. Autentizace klienta. Metodě logon() se předá identifikace session Domain Controlleru a ntlm struktura. Metoda volá funkce SMB a CIFS protokoly, kterými ověří autenticitu klienta.

```
SmbSession.logon( dc, ntlm );
```

5.3.2 Instalace a nastavení

Pokud bude používána knihovna jCIFS přes filter přesměrováním na třídu NtlmHttpFilter, pak je třeba upravit konfigurační soubor web.xml následujícím způsobem.

```
<filter>
   <filter-name>NtlmHttpFilter</filter-name>
   <filter-class>jcifs.http.NtlmHttpFilter</filter-class>
   <init-param>
      <param-name>jcifs.http.domainController</param-name>
      <param-value>192.168.2.1</param-value>
   </init-param>
</filter>

<filter-mapping>
   <filter-name>NtlmHttpFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

Přidáním definice filteru do konfiguračního souboru definujeme název filteru a třídu, která má autentizaci provést. Zároveň se této třídě předávají parametry, ve kterém se definují proměnné. Základní definicí je adresa Domain Controlleru (například ip adresa 192.168.2.1).

6 Kerberos

6.1 Kerberos úvodem

Kerberos je protokol zajišťující bezpečné ověření totožnosti přes nezabezpečené sítě.

Svého účelu dostává protokol Kerberos v síťových aplikacích, od kterých jsou očekávány následující dva základní požadavky. Jednoduchost a bezpečnost autentizace. První požadavek je splněn redukcí uživatelských účtů v systému na jeden a s tím související redukce počtu tajných autentizačních údajů na jeden. Druhý požadavek je splněn komplexní správou účtů založenou na šifrování a bezpečném přenosu zpráv, kdy žádná data neputují po síti v textové podobě.

Kerbeos je autentizační služba [2], která poskytuje bezpečnost, single sign-on, důvěryhodnost a oboustrannost třetích stran. Kerberos je bezpečný protokol. Přenos dat přes síť neprobíhá nikdy v textové podobě. Unikátnost protokolu spočívá především v použití lístků, časově omezených šifrovaných zpráv, které ověřují totožnost uživatele bez toho, aby byly po síti zaslány bezpečnostní data jako uživatelské heslo. Důvěryhodnost je směřována vůči třetí straně centralizovaného autentizačního serveru (např. KDC).

Single sign-on je přístup k řešení autentizace uživatele takový, že po uživateli je požadováno jedno přihlášení do sítě zdrojů, které podporují Kerberos. Toto přihlášení bývá obvyklé při autentizaci do operačního systému.

6.1.1 Vývoj projektu Kerberos

Původ slova

Původ slova Kerberos je převzat z řecké mytologie, kde Kerberos jako trojhlavý pes chránil cestu do podsvětí odkud se duše již nemohly vrátit zpět na zem. Příznačně je jméno síťového protokolu odvozeno právě kvůli těmto vlastnostem, kdy protokol Kerberos ověřuje přístup uživatelů ke zdrojům sítě. Slovo Kerberos je rozšířeno obecně navzdory původnímu mytologickému jménu Cerberus. Je to způsobeno rozdílným hláskováním téhož jména. [3]

Athena projekt

Projekt byl založen v roce 1983. Cílem projektu bylo vyvinout systém pro integraci počítačů do MIT osnov. Od začátku byl projekt koncipován jako síťový systém, založený na bázi klient-server architektury. Základem byl protokol pro síťovou autentizaci. Nový autentizační systém centralizoval důvěrné služby do počítačů, které byly k tomuto účelu úzce specializované, kontrolované a monitorované, a zároveň poskytoval zabezpečený přenos informací mezi těmito centry zabezpečení a ostatními stroji v síti.

Protokol Kerberos nebyl jediným produktem, který vzešel z úsilí projektu Athena. Dalšími významnými byly například X Window System, který je nyní používán jako základ pro Unix GUI. Dalšími síťovými balíčky byly mimo jiné i Hesiod, Moira, MUSE, Discuss a Zephyr. [4]

Verze protokolu Kerberos

Kerberos má již za sebou coby protokol neuvěřitelných 25 let. Po tuto dobu byl různě upravován a vyvíjen a dnešní podoba protokolu je označena jako Kerberos v5. Tomu ale předcházel dlouhý vývoj. Prvními vývojovými verzemi byli především v1, v2 a v3. Tyto verze měly své omezení a byly spíše ve stavu testování a zavádění do stále nových prostředí.

Verze Kerbera v4 je prvním kompletním autentizačním balíkem, jehož první vydání bylo uskutečněno 24.ledna 1989. V té době zároveň velký projekt Andrew File System přijal koncept Kerbera 4 za svůj vlastní autentizační mechanismus, čímž došlo k jeho masivnímu rozšíření. Základy Kerbera 4 byly zahrnuty již v technickém plánu projektu Athena MIT. Tímto byla tedy vize projektu Athena naplněna. Kvůli DES šifrování však organizace mimo USA neměly možnost stahovat tuto verzi Kerbera [2]. Z toho důvodu vývojáři Kerbera vytvořili speciální verzi, která se začala ujímat i za hranicemi USA. Dnes ještě stále několik distribucí Kerbera 4 existuje (např. kth-krb).

Kerberos 5 byl vyvinut tak, aby přidal nové bezpečnostní rysy, které ve verzi 4 chyběly. Jedná se zatím o poslední verzi a je celá dokumentovaná v RFC 1510. Základními novými rysy jsou například doporučené zasílání a delegování, opakovaná paměť, více flexibilní autentizace napříč domén, rozšířené typy šifrování a další.

6.2 Základy protokolu Kerberos

6.2.1 Tři A

Koncept Kerbera je založen na třech pilířích, které jsou mezi síťovými profesionály známy jako tři A. Autentizace, Autorizace a Audit. [2] K těmto patří prostředí, ve kterých Kerberos operuje, a těmi jsou úložiště informací. Mezi nejznámější a protokolem nejvíce využívané patří zejména LDAP. Základ všech těchto přístupů tvoří bezpečnost, míra utajení a z toho vyplývající integrita zpráv.

Autentizace

Autentizace je proces ověření identity daného uživatele [5]. Pro ověření identity je uživatel tázán na předem domluvenou informaci. Tato informace může spadat do jedné z těchto k kategorií informace [5]:

1. co uživatel zná – heslo, PIN
2. co uživatel má – USB dongle, smart card, privátní klíč
3. čím uživatel je – otisk prstu, snímek oční zornice

Jak je vidět z uvedeného výčtu, heslo nebo pin je jen jednou z možností autentizace. V tomto pohledu Kerberos přeskočil svou domu. Obdobná řešení single sign-on založená například na protokolu NTLM se nedají, nebo jen velmi těžko, přizpůsobit dnes novým přístupům v ověření uživatele založeném například na snímání otisku prstu. V tom je Kerberos a jeho systém lístků aktuální.

Na systému Kerberos je cenné to, že není třeba striktně specifikovat, který způsob autentizace bude použit.Může se použít jak RSA SecureID, biometrické snímání, tak heslo jako textové pole. Například Microsoftí Windows 2000 a vyšší podporují autentizaci uživatele Kerberem pomocí čipových karet, které poskytují jednodušší postup při logonání pro uživatele a zároveň poskytují bezpečnější mechanismus ověřování uživatele [2].

Autorizace

Autorizace je postup, který omezuje přístup k informacím, funkcím a dalším objektům [5]. Základem ověřování přístupu jsou ACL (access control list), které asociují jednotlivé uživatele s danými jim přiřazenými právy.

Důležitá je bezpečnostní implikace autorizace. Ta je bezpečná pouze tehdy, pokud je uživatel validně autentizován. Autorizace je zastavena v případě, že autentizační metoda nevytváří důvěrné spojení s autentizační autoritou.

Audit

Audit není preventivní část Kerbera (jako výše uvedené), ale jedná se o reaktivní část. Systém logování vestavěný do Kerbera umožňuje dobré možnosti pro ověřování správné funkčnosti autentizace a na ni navazující autorizace.Každá implementace Kerbera poskytuje dobrá řešení pro logování akcí.

6.2.2 Uložiště

Obecně se často tvrdí, že adresáře jako Unix /etc/passwd, NIS, LDAP a další poskytují dobrou autentizaci systému sami o sobě. Není tomu tak. Úložiště mohou obsahovat a popisovat síťové objekty jako tiskárna nebo počítač. Ty jsou často uloženy jako jednoduchý text. LDAP nebo Actice Directory obsahují komplexní adresářovou strukturu. Autentizace proti adresáři může mít dva tvary [2]. Prvním možností je, že klinstský počítač kontaktuje adresář a obdrží hash uživatelova hesla, kterou u sebe zkontroluje. Druhá možnost je vidět například u LDAP, kdy uživatel ověří svá práva tím, že zkusí přistoupit ke službám adresáře pomocí svých údajů. Zde mohou být použity protokoly jako Samba nebo PAM modul. Použití Kerbera je lepší v několika ohledech. Zejména použitím lístků může uživatel přistupovat ke všem možným zdrojům na síti bez kolování uživatelova hesla po síti. Kerberos lístky jsou šifrované zprávy, které jsou validní pouze po určitou dobu (typicky 8-24 hodin). Tímto mechanismem není nikdy posláno heslo po síti jako jednoduchý text.

6.2.3 Utajení a kryptování

Šifrování (jako cryptography, cryptos – tajný, graphein – psaní) poskytuje funkčnost pro zašifrování a rozšifrování zpráv posílaných po síti.Kerberos nepoužívá šifrování pouze pro ochranu autentizačních údajů zaslaných a obdržených objekty v síti, ale také poskytuje ochranu proti neoprávněnému podvrhu zprávy zvenku sítě, kdy je snaha vytvořit a podstrčit do sítě zprávu podobnou těm, které v této síti kolují. Kerberos obsahuje několik možností šifrování dat[2]. Nejrozšířenějšími standardy šifrování Kerbera jsou DES a AES (Advance Encryption Standard). Dalším široce rozšířeným algoritmem pro šifrování je RC4, který je široce rozšířen u Microsoft implementacích Kerbera.

Integrita zpráv je zaručena hash funckemi. Funkce je matematická, jednosměrná a bere rozdílně dlouhý vstup a vytváří z něj výstup o pevně dané velikosti (64-256 bitů). Mezi takové algoritmy patří u Kerbera mimo jiné CRC-32, MD5 a SHA1 (Secure Hash Algorithm) [2].

6.3 Terminologie

6.3.1 Doména

Jak WebKDC, tak i všechny aplikační servery mají svůj vlastní principal v Kerberos systému. Tento fakt pomáhá zvyšovat bezpečnost celého systému, neboť je možné definovat seznam "povolených" aplikačních serverů, které smí požádat o identitu přistupujícího uživatele [6]. Každí principal je definován long-term klíčem. Tímto klíčem může být například heslo. Principal má v síti globálně unikátní jméno a je začleněn do hierarchické struktury. Každý principal začíná uživatelským jménem a názvem služby. Poté následuje volitelná položka. Tento principal pak značí unikátní identifikaci v rámci domény. Kerberos definuje administrativní kontrolní doménu, která je rozdílná od jakékoliv jiné Kerberos instalace [2]. Podle konvence Kerberos doména v rámci DNS domény je tato doména převedená na velká písmena. Např. DNS doména – necas.cz, Kerberos doména NECAS.CZ. Tato konvence ale není podmínkou.

Jako příklad uvedeme Kerberos principal, který je spojen s Tomášem Nečasem, který pracuje v IT oddělení firmy Deny:

tne@IT.DENY.CZ

Tento tvar je validní jak v Kerberos verzi 4 tak 5.

6.3.2 Klíče a hesla

Všechny klíče jsou sdíleny alespoň dvěmi stranami. Jednou stranou je požadovaná služba, nebo uživatel, a druhou stranou je Key Distribution Center (dále KDC). Kerberos využívá funkci string2key , která má na starosti konverzi uživatelského hesla na kryptovací klíč. Tato funkce aplikuje

několik transformací během kterých se znakově založené uživatelské heslo transformuje na čísel, které vytvoří kryptovací klíč [2].

Nejdůležitější součástí transformace je tzvn. „salt“. Obecně řečeno je salt posloupností znaků, která je přidána do hesla před tím, než se z něj udělá hash. Pro Kerberos 5 je salt zadáván ve formě Kerberos domény [7]:

Kpippo=string2key(Ppippo+"pippo@EXAMPLE.COM")

Kpippo je kryptovací klíč uživatele pippo a Ppipo je nešifrované heslo daného uživatele. Pro kompatibilitu s Kerberos 4 verzí je nastaven salt na null.

6.3.3 KDC

Autentizační server v Kerberos doméně, který je založen na funkci distribuci lístků pro přístup k různým službám, se nazývá Key Distribution Center [7]. Jako fyzický server jej můžeme logicky členit na následující tři části:

1. Databáze – kontainer položek uživatelů a služeb označených jako principals
2. Autentizační Server (AS) – autentizační služba při přihlášení uživatele do Kerberos domény
3. Ticket Granting Server (TGS) – služba pro vydávání lístků na vyžádání autentizovaným uživatelům a službám

Tyti tři služby, ač logicky nezávislé, jsou implementovány obvykle v jednom programu a běží společně v jednom místě procesu. V jedné Kerberos doméně musí existovat alespoň jeden KDC.

KDC software obsahuje informaci pro každý principal v bázi, jako platnost hesla, poslední změnu hesla a další [7]. Windows 2000 a vyšší ukládá tyto informace v databázi Active Directory. Otevřené implementace jako MIT ukládají data do speciálních bází postavených na KDC souborovém systému [2].

6.3.4 Lístky

Lístek je kryptovaná datová struktura vydaná KDC serverem a sdílející kryptovaný klíč, který je unikátní pro každou session v rámci Kerberos domény, a některé další příznaky a pravidla pro pohyb lístku v dané Kerberos doméně [7].

Lístek souží obecně dvěma účelům [2]:

1. potvrzení identity koncového účastníka komunikace
2. ustanovení kryptovacího klíče, které obě strany komunikace sdílí pro bezpečnou komunikaci (nazýván i jako session key)

Každý lístek obsahuje jméno principal (požadujícího), jméno principal (služby), délka platnosti lístku, seznam IP adres, ze kterých je možné lístek použít, tajné sdílené session key pro bezpečnou komunikaci s daným zdrojem [3].

Lístky jsou uchováván v paměti, která je rozdílná podle typu a implementace Kerbera. Pokud budeme brát v úvahu defaultní úložiště lístků, můžeme si pomocí příkazu klist vypsat základní data souboru s lístky [7]:

$ klist

Ticket cache: FILE:/tmp/krb5cc_500

Default principal: pippo@NECAS.CZ

První částí výpisu kinit je uživatelův principal, který je uložen v souboru /tmp/krb5cc_500. Následují služby a jejich principaly s daty jako Valid starting, Expired a Service principal.

6.4 Protokoly

V nynější době existují dva rozšířené protokoly Kerberos v4 a v5. Kapitola si všímá detailů těchto protokolů. Zatímco koncept a návrh protokolů je u obou verzí velmi podobný, velké rozdíly jsou patrné v základech protokolů a jejich implementaci [2]. Základní operace protokolu Kerberos byly publikovány v roce 1978 pány Needham a Schroeder. Tehdy se tento princip nazýval Needham and Schroeder protokol.

6.4.1 Needham a Schroeder protokol

Protokol byl navržen jako bezpečnostní atentizační systém. Práce, které se stala v roce 1978 základem dalšího vývoje, popisuje dva rozdílné protokoly. První z nich popisuje použití privátního kryptovaného klíče a právě tento protokol je základem protokolu Kerberos.

Protokol Needham-Schroeder definuje tři účastníky protokolu [2]: klientský stroj, tázaný server a autentizační server. Tázaný server je stroj, na který chce klient přistoupit. Jedná se o aplikační server, který poskytuje službu, ke které chce klient přistoupit. Autentizační server je stroj, který obsahuje kopie kryptovacích klíčů uživatelů a služeb na síti.

Needham a Schroeder protokol poskytuje mechanismus pro bezpečnou distribuci šifrovacích klíčů na klienta i na server. Protokol začne kontaktováním autentizačního serveru a zasláním zprávy obsahující identity obou stran a tzv. nonce jako vygenerovanou hodnotu. Autentizační server vrací kopii session klíče, jméno aplikace, původní nonce, aplikační kopii session klíče, jméno klienta, aplikační klíč a uživatelský klíč. Jednotlivé položky odeslané zprávy jsou vnořeny tak, aby se daly dále přeposílat. Po obdržení zpávy klientem ověří klient správnost doručeného balíku a vyjme z něj session klíč, jméno klienta a aplikační klíč. Tyto hodnoty pak pošle aplikačními serveru [17].

Aplikační server potvrdí spojení tím, že zvýší vygenerované nonce klientem o jedničku a spolu s session klíčem přepošle nazpět klientovi.

Jak uvidíme dále, velkou většinu a celý základ Needham-Schroederova protokolu přebral Kerberos protokol.

6.4.2 Kerberos v4

Vznik protokolu

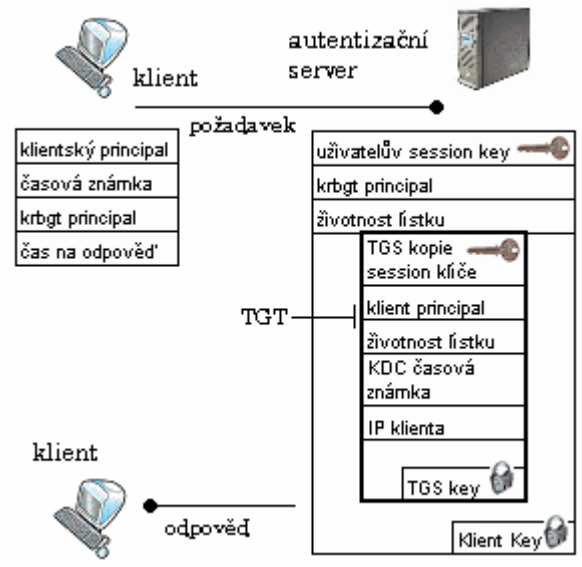
Jedná se o protokol založený na základech Needham-Schroeder protokolu se dvěma základními rozdíly [2].

Prvním rozdílem je omezení množství odesílaných zpráv mezi klientem a autentizačním serverem. Kerberos 4 protokol opakování zprávy je zmařeno díky autentizační zprávě, která je vytvořena na základě lokálního času klienta kryptovaného nově vyjednaným session klíčem daného spojení.

Druhým rozdílem je vytvoření konceptu TGT - Ticket Granting Ticket. Tento koncept umožňuje klientovi autentizovat se vůči většímu počtu aplikačních služeb. To je i hlavní výhodou Kerbera – vytvoření single sign-on přístupů k službám v dané doméně, aniž by bylo nutné stálé ověřování. Jako důsledek této politiky se dělí Kerberos služba na část AS (Autentizační Server) a TGS (Ticket Granting Server).

Autentizační server

Server má jednu základní funkci. Obdržet požadavek obsahující uživatelské jméno klienta, který požaduje autentizaci a vrátit šifrovaný lístek TGT pro daného uživatele [2]. Poté může klient použít lístek TGT na přihlášení se do dalších aplikací a služeb. Celý proces komunikace klienta s AS je vidět na obrázku 1.



Obrázek 16: Autentizační služba AS serveru [2]

První zpráva putuje od klienta k AS a je známa jako AS_REQ. Zpráva je poslána jako jednoduchý text a obsahuje identitu klienta, lokální čas klienta a principal TGS. Dle konvence je TGS u Kerbera 4 pojmenován krbtgt.

Po obdržení zprávy AS_REQ od klienta AS ověří, zda zadané principals existují a zkontroluje čas na zprávě (tolerance obvykle 5 minut). Poté generuje AS session klíč. Tento klíč bude sdílet klient s TGS serverem. AS vytvoří dvě kopie session klíče a jeden pošle v rámci TGT zpět klientovi a druhý pošle TGS serveru.

Odpovědí na AS_REQ je zpráva AS_REP. Tato odpověď se skládá ze dvou částí. První část obsahuje základní data pro doručení nazpět klientovi, který o danou službu usiluje. Tato část je uzamknuta pomocí klientského klíče, který sdílí pomocí nastavení administrátora daný klient s autentizačním serverem. Druhou částí je tzv. TGT – Ticket Granting Ticket. Tento lístek obsahuje data potřebná k tomu, aby si daný klient mohl zažádat o přístup k určité službě v rámci Kerberos domény. Ten získá na základě obdrženého session klíče a TGT lístku dotazem na TGS s žádostí o příslušný lístek relace mezi klientem a požadovanou službou. TGT lístek je uzamčen TGS klíčem, který zná pouze TGS server a použije jej pokud při příchodu TGT lístku s žádostí o přidělení určité služby.

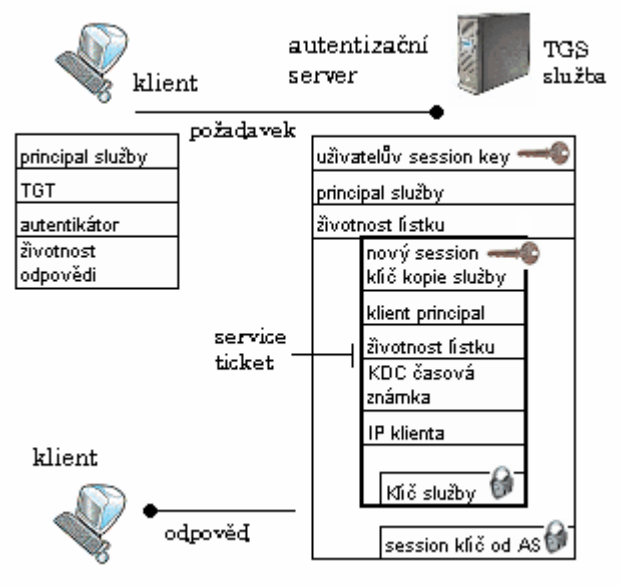
Když klient obdrží zprávu, dešifruje ji pomocí klientského klíče (obvykle heslo). Pokud proběhlo dešifrování v pořádku, má klient potřebné náležitosti k tomu, aby zažádal o povolení přístupu k určité službě. Klient zároveň nemůže číst TDT lístek. Ten je uložen v pověřené paměti.

Ticket Granting Server

Pokud uživatel potřebuje přistoupit k určité autentizované službě v rámci Kerberos domény, musí k této akci použít příslušný lístek. Ten obdrží od TGS na základě TGT. Klient tedy musí připravit zprávu, která bude obsahovat tyto tři části: TGS požadavek (request), kopii TGT lístku obdrženého od AS a autentikátor.

Autentikátor se skládá z časové známky, která je kryptovaná session klíčem. Autentikátor zabezpečuje, že každý lístek je unikátní, a ověřuje, že klient zná sdílený session klíč ustanovený AS serverem. Bez autentikátoru by případný útočník mohl poslouchat v rámci sítě a udělat kopii TGT, která by mu pak umožnila případný přístup k TGS nebo jinému zdroji v rámci domény.

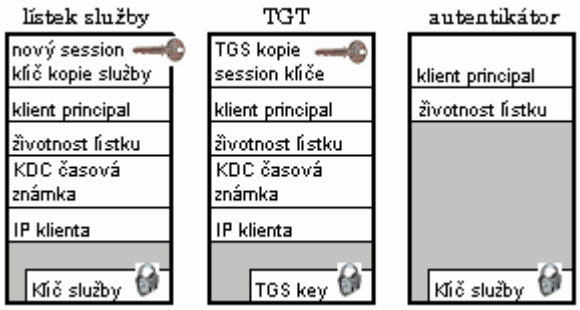
Na základě požadavku TGS_REQ je na straně KDC generována zpráva zvaná TGS_REP [3]. Tato zpráva zahrnuje novou množinu session klíčů, sdílených mezi klientem a aplikačním serverem. Klientská kopie session klíče je ke kryptovaná starší verzí session klíče [2]. Kopie nové session pro komunikaci klienta s požadovanou službou je vestavěna uvnitř lístku služby (service ticket) a je kryptována pomocí long-term klíče dané služby. Tento klíč je nastaven administrátorem napevno a je unikátní v dané doméně. Po obdržení těchto informací si klient přidá service ticket a nový session klíč do své pověřené paměti pro pozdější použití. Celý systém komunikace s KDS je vidět na obrázku 2.



Obrázek 17: Autentizační služba TGS serveru [2]

Na obrázku 3 jsou pro zjednodušení vidět všechny tři možné lístky, které kolují mezi KDC a klientem.

Poslední etapou v procesu přístupu k autentizované službě je zaslání service lístku dané službě, která pomocí klíče dané služby rozšifruje obsah, ustanoví session key a zkontroluje časové příznaky pro ověření platnosti příchozího lístku.



Obrázek 18: Kerberos lístky [2]

Transformace řetězce na 56-bitové klíče, které jsou pak použity pro šifrování a dešifrování zpráv v rámci Kerbera, odkazuje na string-to-key funkci. Tato funkce je jednosměrnou fnukcí, která generuje hash a pro vstup používá v případě Kerbera 4 heslo a výstupem je 56-bitový hexadecimální DES klíč [2]. Matematicky je velmi obtížné nazpět získat příslušné heslo, nicméně v dnešní době je právě z tohoto hlediska vhodnější používat Kerberos 5 s pokročilejšími technikami jednosměrného šifrování.

6.4.3 Kerberos v5

Kerberos 5 je evoluční nástupce Kerbera 4. Kerberos 5 si ponechává veškerou funkcionalitu protokolu Kerberos 4, ale přidává k němu jistá rozšíření. Z hlediska implementace se však jedná o téměř nový protokol.

V první řadě Kerberos 5 posiluje bezpečnost v oblasti jednosměrných DES funkcí. Dalším novým rysem je podpora delegování a přeposílání v rámci Kerberos domén. Přeposílání údajů uživatele (úložišť s lístky) umožňuje uživateli přistupovat ke službám na vzdáleném serveru. Tento nový jev funguje tím způsobem, že při přihlášení uživatele a komunikaci s AS se TGT bezpečně přepošle i na vzdálené servery. Pro umožnění této funkcionality na bázi komplexity a multiplatformní charakteristiky vývojáři Kerbera vybrali technologii zvanou ASN.1 k popisu protokolu Kerberos 4. ASN.1 umožňuje vývojářům protokolu vytvářet protokoly na bázi abstraktního jazyka [2].

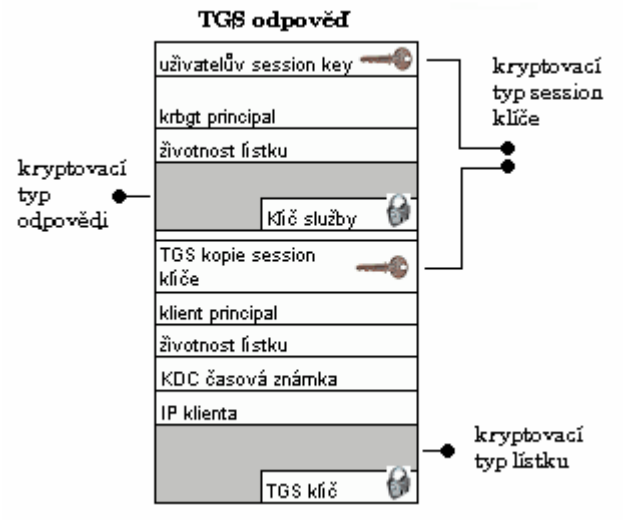
Kvůli zpětné kompatibilitě poskytuje Kerberos 5 transformační službu zvanou krb524, která provádí transformaci lístků Kerbera 5 na lístky Kerbera 4.

AS a TGS

V rámci AS došlo k několika změn oproti Kerberos 4. Na vrstvě protokolu je změněno několik rysů protokolu, mezi které patří použití ASN.1, dvojité šifrování na straně AS a TGS v rámci KDC.

Nové násobné šifrování podporované protokolem verze 5 znamená, že může být použito při transakcích protokolu více šifrovacích typů. Oddělené kryptovací typy mohou být použity v různých zprávách, jakými jsou Lístek, Odpověď, Session klíč. Následující diagram ukazuje, kde jsou umístěny jednotlivé kryptovací typy (obrázek 4.)

Dalším rozdílem oproti Kerberu verze 4 jsou lístky, které mají možnost voleb rozšířených rysů [2]. Tyto příznaky mohou být například následující: možnost předávat lístky, nastavení proxy lístků, obnovitelné vlastnosti lístků a možnost nastavit platnost lístků až po určitém datu (tzv. postdated tickets).

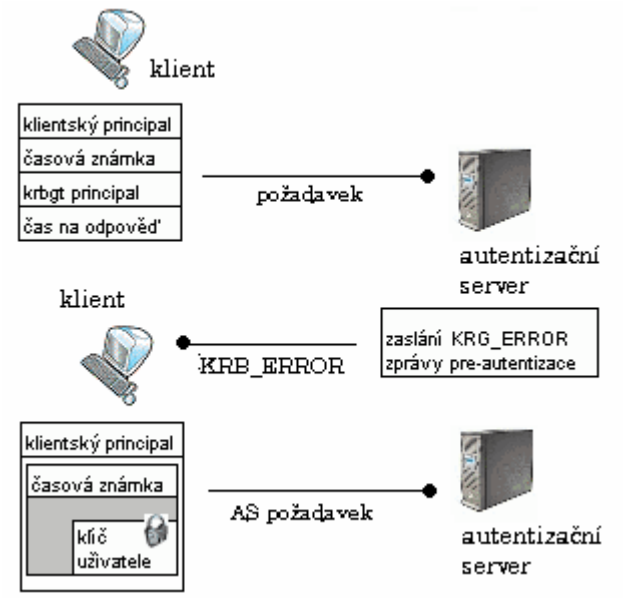


Obrázek 19: Kryptovací typy [2]

Pre-Autentizace

Další z nových rysů protokolu Kerberos 5. Základem vzniku tohoto mechanismu v nové verzi protokolu byly možné útoky v předcházející verzi, které měli za cíl zaslání podvrženého klíče daného

principal KDC serveru. Aby se tyto útoky znesnadnily, byl zaveden žadatel, který ověřuje identitu ještě před tím, než se zašle důvěrná informace v podobě TGT lístku zpět klientovi. Existuje několik typů pre-autentizace, nicméně nejrozšířenější je metoda časové známky (pa-enc-timestamp) [2].



Obrázek 20: Proces pre-autentizace [2]

Proces pre-autentizace je popsán nejlépe na obrázku 5. Pokud klient zažádá o TGT lístek a KDC má nastavenu pre-autentizaci, tak zašle nazpět klientovi KRB_ERROR zprávu místo AS_REP. Klient na to generuje požadovaná data, nejčastěji časovou známku. Pokud jsou data KDC serverem akceptována, pak pošle klientovi požadovaný lístek.

6.5 KDC

Existuje několik implementací serveru KDC a protokolu Kerberos. Mezi nejznámější patří například MIT, Heimdal nebo Windows domain controller. Původní Kerberos (z laboratoří MIT) měl omezený přístup na evropský trh i jinde do světa, protože platily přísné pravidla omezeného exportu šifrovacích algoritmů ze Spojených států. Proto vzniká evropský, volně šiřitelný klon s názvem Heimdal. I když má Heimdal jiný kód, podporuje API MIT Kerbera. Heimdal je integrovatelný s více operačními systémy včetně BSD [1].

Windows domain controller podporuje pouze Kerberos verze 5 a nepodporuje na rozdíl od MIT nebo Heimdal žádnou zpětnou kompatibilitu s protokolem verze 4. Domain controller podporuje pouze RC4 kryptování a starší typy DES.

7 Aplikační podpora protokolů

V kapitole bude popsána úroveň podpory autentizace v prohlížečích (IE, Firefox, Opera) a na různých operačních sytsémech (Windows, Linux). Dále bude následovat popis podpory autentizace na webových serverech (Apache, Java aplikační servery, IIS).

7.1 Úroveň podpory klienta

Podpora autentizace v prohlížečích pro Kerberos a NTLM.

protokol	**Kerberos**	**NTLM**
MS Windows		
MSIE 5.x starší	NE	ANO
MSIE 5.x a 6 +	ANO	ANO
Mozilla Suite 1.7.5 +	ANO	ANO
Mozilla Firefox 1.0 +	ANO	ANO
Netscape 7.2 +	NE	ANO
Netscape 8.0 +	ANO	ANO
Opera starší 9.x	NE	NE
Opera 9.x a novější	NE	ANO
Linux		
Mozilla Suite 1.7.5 +	ANO	
Mozilla Firefox 1.0 +	ANO	
Netscape 7.2 +	ANO	
Konqueror 3.5 +	ANO	
MacOS X		
Mozilla Suite 1.7.5 +	ANO	
Mozilla Firefox 1.0 +	ANO	
Netscape 7.2 +	ANO	
Safari OSX 10.4.3	ANO	

7.2 Úroveň podpory serveru

7.2.1 Apache

Webový Server Apache obsahuje podporu protokolu Kerberos formou přídavného modulu mod_auth_kerb. Použitím Basic Authentication mechanismu obdrží uživatelské jméno a heslo od prohlížeče a ověří jej oproti Kerberos Serveru podle toho, jak je ověření nastaveno. Modul dále podporuje Negotiation authentication metodu, která zároveň zaručuje plnou podporu Kerberos autentizace založené na zasílání lístků a nepožaduje tak po uživatelovi zadat heslo do formuláře. Pokud se používá Negotiate metoda, je třeba podpora ze strany webového prohlížeče (viz. Úroveň podpory klienta).

Modul podporuje jak Kerberos v4 tak Kerberos v5 protokol pro ověření hesla. Negotiate mechanismus může být použit pouze s Kerberos v5. Modul podporuje jak 1.x tak i 2.x verzi Apache.

Při použití Basic Auth mechanismu, nepoužívá modul žádné speciální kryptovací techniky. Při použití metody Negotiate je použito kódování Base64. Je tedy jednoduché převést řetězec na jednoduchý text. Abychom tomuto zabránili, je výhodné používat modul mod-ssl nebo Apache-ssl [13].

Podpora protokolu NTLM je zaručena dvěma moduly. Prvním modulem je mod_ntlm. Tento modul však nepodporuje protokol NTLM2, který je už jako jediný podporován v rámci Windows Vista. Odpora NTLM2 je umožněna modulem mod_auth_ntlm_winbind. Tato řešení jsou využívána na serveru Apache bez použití Windows. Pokud je použit OS Windows, je řešením modul mod_auth_sspi [14].

7.2.2 IIS

Microsoft Internet Information Services (IIS) podporuje jak Kerberos tak NTLM protokol pro síťovou autentizaci. IIS přímo zasílá Negotiate hlavičku během Windows autentizace. Hlavička tak dá klientovi vybrat, zda si chce zvolit mezi Kerberos a NTLM autentizací. Negotiate proces nevybere Kerberos autentizaci pokud platí alespoň jedna z následujících možností:

- Jeden ze systémů, které jsou zapojeny do autentizace, nemohou použítKerberos autentizaci.
- Volaná aplikace neposkytuje informace pro použití Kerberos protokolu.

Pro umožnění procesu Negotiate k výběru Kerberos protokolupro síťovou autentizaci, musí klientská aplikace poskytovat Service Principal Name (SPN) a User Principal Name (UPN), nebo NetBIOS jméno účtu jako cílové jméno. V opačném případě Negotiate proces vybere NTLM protokol jako autentizační metodu.

Pro nastavení použití Kerberos a NTLM protokolu je třeba potvrdit, že Negotiate hlavička je nastavena v NTAuthenticationProvider vlastnostech. Tyto vlastnosti se nastavují rozdílně podle verze IIS[15].

Pro nastavení autentizace na Kerberos a NTLM u IIS v6.0 je třeba zjistit aktuální hodnoty NTAuthenticationProvider. To je možné provést následujícím příkazem :

```
cscript adsutil.vbs get w3svc/WebSiteIDnumber/root/NTAuthenticationProviders
```

Adsutils.vbs najdeme na C:/Inetpub/Adminscripts. Pokud je proces Negotiate podporován, vrátí příkaz následující hodnotu:

NTAuthenticationProviders : (STRING) "Negotiate,NTLM"

Pokud takovou hodnotu Server nevrátí, je třeba zadat následující příkaz pro nastavení autentizace:

```
cscript adsutil.vbs set w3svc/ WebSiteIDnumber /root/NTAuthenticationProviders "Negotiate,NTLM"
```

7.2.3 Java Aplikační Servery

Aplikační Servery postavené na platformě J2EE jsou odkázány v případě řešení Single Sign-on jednak na podporu autentizace u J2EE a na produkty třetích stran, které poskytují různá řešení a moduly pro autentizaci přes Kerberos a NTLM.

Podpora u J2EE je umožněna komponentou JAAS (Java Authentication Service). Jedná se o balík integrovaný od verze J2SDK 1.4. JAAS je implementací standardního modulu PAM (Pluggable Authrntication Modul).

Komponentu JAAS pak využívá několik možných implementací autentizace na bázi Javy. Mimo jiné můžeme jmenovat například následující projekty:

- **JOSSO** (Java Open SSO). Založeno na JAAS, Web Services, EJB, Struct, Servlety a JSP. Běží na JBoss 3.2.6. aplikačním serveru a Jakarta Tomcat 5.0. Podporuje LDAP a konfiguraci vícevrstvých Single Sign-on konfigurací.
- **JBoss NegotiateKerberos**. Řešení používá Negotiate, Kerberos a NTLM.
- **JGSS** IBM. Řešení na základě JAAS a GSS.
- **jCIFS**. Java balík, který obsahuje podporu NTLM autentizace a ověřování přes SMB/CIFS protokol. V rámci rozšíření jCIFS-ext je zabudovaná podpora Kerberos autentizace.

Více o podpoře protokolů Kerberos a NTLM z prostředí aplikačních serverů J2EE bude následovat ve zbývajících kapitolách implementace Kerberos protokolu pro ověření klienta v rámci diplomové práce.

Závěr

Single Sign-on je řešení, které zajímá v dnešní době nejen teoretiky informačních technologií, ale i IT veřejnost. Možnost doladit aplikaci tak, aby byla bezpečnější a zároveň pro uživatele i administrátora jednodušší, je velice lákavá.

Dokument čeká rozšíření v podobě implementace protokolu Kerberos na architekturu WEBACS. Rozšíření bude obsahovat jak část teoretickou, která bude analyzovat možné přístupy k implementaci protokolu Kerberos, tak část praktickou, která bude obsahovat implementaci modulů pro architekturu WEBACS.

Závěrem tohoto dokumentu o možnostech, které poskytuje řešení SSO, bych se rád pozastavil nad jednou skutečností, která mě při psaní zaujala. Jak bylo v textu napsáno, protokol Kerberos je protokolem velice starým. Napadá mě tedy otázka jak je možné, že tak starý protokol se stále těší veliké oblibě, jak je možné, že zájem o možnosti Kerberos SSO stále roste? Myslím, že je to dáno jednoduchou a přitom dokonalou koncepcí tohoto protokolu. A právě taková řešení mají v business sféře své místo i budoucnost.

Literatura

[1] Radkovič,P.: Kerberos a PAM [online]. [cit.10.5.2007]. CZ, Dostupné na URL: http://www.fi.muni.cz/~kas/p090/referaty/2005-jaro/ct/radkovic_KrbPAM.html

[2] Garman,J.: *Kerberos The Definitive Guide* USA, Sebastopol CA Oreilly, 2003, s.6-86. ISBN 0-596-00403-6

[3] Kerberos_(protokol) [online]. Poslední modifikace 12.prosince 2005. [cit.10.5.2007]. USA, Dostupné na URL: http://www.tvwiki.tv/wiki/Kerberos_(protocol)

[4] Kohl,J.: The Kerberos Network Authentication Service [online]. Poslední modifikace září 1993. [cit.11.5.2007]. USA, Dostupné na URL: http://www.ietf.org/rfc/rfc1510.txt

[5] Kerberos_(protokol) [online]. Poslední modifikace 12.prosince 2005. [cit.11.5.2007]. USA, Dostupné na URL: http://www.tvwiki.tv/wiki/Kerberos_(protocol)

[6] Glolmus,P.: Web single sign-on řešení pro web [online]. Poslední modifikace 30.listopad 2005. [cit.11.5.2007]. CZ, Dostupné na URL: http://www.cesnet.cz/doc/techzpravy/2005/webiso/

[7] Ricciardi,F.:The Kerberos protokol [online]. Poslední modifikace 26.listopad 2006. [cit.11.5.2007]. IT, Dostupné na URL:http://www.zeroshell.net/eng/kerberos/Kerberos-definitions/#1.3.5

[8] Single Sign-on [online]. Poslední modifikace 27.prosinec 2007. [cit.31.12.2007]. EN, Dostupné na URL: http://en.wikipedia.org/wiki/Single_sign_on

[9] SSO [online]. Poslední modifikace 2005. [cit.11.5.2007]. EN, Dostupné na URL: http://wiki.java.net/bin/view/Javapedia/SSO

[10] Loshin,P.: Konec zapomenutých hesel: Single Sign-on [online]. Computerworld, rok vydání 2001, číslo vydání 44. [cit.12.5.2007]. CZ, Distupné na URL: http://archiv.computerworld.cz/cwarchiv.nsf/clanky/7DB09143FB84FBB0C1256B480047A838?OpenDocument

[11] Glolmus,P.: Web single sign-on řešení pro web [online]. Poslední modifikace 30.listopad 2005. [cit.11.5.2007]. EN, Distupné na URL: http://artax.karlin.mff.cuni.cz/~brain/diplomka.pdf

[12] RSA security [online]. Poslední modifikace 2007. [cit.31.12.2007]. EN, Dostupné na URL: http://www.tsoft.cz/index.php?q=node/312/print

[13] Kerberos Module for Apache [online]. [cit.30.12.2007]. CZ, Dostupné na URL: http://modauthkerb.sourceforge.net/

[14] Gal,A.: NTLM auth module for Apache [online]. Poslední modifikace 2000. [cit.29.12.2007]. EN, Distupné na URL: http://modntlm.sourceforge.net/

[15] How to configure IIS to support Kerberos and NTLM protocol [online]. Poslední modifikace 19.ledna 2007. [cit.29.12.2007]. EN, Dostupné na URL: http://support.microsoft.com/kb/215383

[16] Java Security [online]. Poslední modifikace 2007. [cit.28.11.2007]. EN, Dostupné na URL: http://java.sun.com/javase/6/docs/technotes/guides/security

[17] Needham and Schroeder protokol [online]. Poslední modifikace 24.březen 2007. [cit.12.5.2007]. USA, Distupné na URL: http://en.wikipedia.org/wiki/Needham-Schroeder

[18] Hunting,G.: Authentication World [online]. [cit.30.12.2007]. EN, Dostupné na URL: http://www.authenticationworld.com/Single-Sign-On-Authentication/

[19] Introduction to SSL [online]. Poslední modifikace 10.9 1998. [cit.5.1.2008]. EN, Dostupné na URL: http://docs.sun.com/source/816-6156-10/contents.htm