VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

ANALÝZA PROVOZU PROTOKOLŮ KERBEROS, NTLM, SAML 2.0

TRAFFIC ANALYSIS OF NETWORK PROTOCOLS KERBEROS, NTLM, AND SAML 2.0

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL KRŮL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. PETER TISOVČÍK**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Krůl Michal**
Program: Informační technologie
Název: **Analýza provozu protokolů Kerberos, NTLM, SAML 2.0**
Traffic Analysis of Network Protocols Kerberos, NTLM, and SAML 2.0
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se s možnostmi monitorování síťového provozu pomocí Flowmon sond a kolektoru.
2. Nastudujte protokoly Kerberos, NTLM a SAML 2.0 a analyzujte možnosti monitorování těchto protokolů v rámci Flowmon sond a kolektoru.
3. Seznamte se s možnostmi útoků, které využívají nebo přímo ovlivňují protokoly Kerberos, NTLM a SAML 2.0.
4. Proveďte simulaci útoků na autentizační protokoly a zkuste potvrdit nebo vyvrátit, že je možné vybrané útoky na úrovni sítě detekovat.
5. Navrhněte a popište několik příkladů monitorování autentizačních protokolů (provozní monitoring, detekce útoku, chybná nebo přímo nebezpečná konfigurace protokolu).
6. Proveďte implementaci monitorování uvedených protokolů jako samostatný modul (plugin) do Flowmon sondy.
7. Ověřte vytvořenou implementaci na zachycených vzorcích síťového provozu a na reálné síti.
8. V závěru práce diskutujte dosažené výsledky a navrhněte další směry pokračování práce.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 4 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Tisovčík Peter, Ing.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 25. října 2019

Abstrakt

Tato práce se zabývá problematikou analýzy a detekce napadení bezpečnostních protokolů v prostředí síťových architektur, jaké se vyskytují například ve velkých firmách. V práci je k problému přistupováno z hlediska analýzy síťového provozu. Hlavní náplní práce je simulace útoků na síťové architektury, kde autentizaci obsluhují zmíněné protokoly, a snaha o jejich detekci pomocí sledování síťového toku. Výsledkem práce je návrh, jak automaticky detekovat útoky v síťových strukturách a plugin pro exportém Flowmon sondy, produktu firmy Flowmon Networks, který bude data potřebná pro provedení této detekce sbírat.

Abstract

This thesis engages the problem consisting of analysis and detection of the attacks carried out on the authentication protocols in the environment of network structures, like those used in big corporations. In this thesis, the problem is examined in the light of the netflow analysis. Main content of the thesis is a simulation of the attacks targeting network architectures, where the authentication is served by mentioned protocols, and effort to detect these attack by the netflow monitoring. The outcome of this thesis is a draft, how to automatically detect the attacks carried out in the network structures, and plugin for the exporter of the Flowmon sond, the product of Flowmon Networks company, which will be extracting the information needed for the performance of the detection.

Klíčová slova

síťový tok, bezpečnostní protokol, Kerberos, NTLM, SAML 2.0, FlowMon, exportér, sonda

Keywords

netflow, authentication protocol, Kerberos, NTLM, SAML 2.0, FlowMon, exporter, sond

Citace

KRŮL, Michal. *Analýza provozu protokolů Kerberos, NTLM, SAML 2.0.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Peter Tisovčík

Analýza provozu protokolů Kerberos, NTLM, SAML 2.0

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Tisovčíka. Zadání této práce mi poskytla firma Flowmon Networks. Výpočetní zdroje pro vytvoření umělého síťového prostředí mi poskytla organizace CESNET MetaCentrum VO. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Krůl
27. května 2020

Poděkování

Tímto bych chtěl poděkovat Ing. Petrovi Tisovčíkovi z Fakulty informačních technologií Vysokého učení technického v Brně za vedení práce a věcné připomínky k práci, firmě Flowmon Networks a.s. a Tomáši Vlachovi za zadání práce a možnost pracovat na tomto tématu, Mgr. Ondřeji Kozákovi z firmy Flowmon Networks a.s. za dohled nad prací, praktické rady k práci a poskytnutí prostředků k vypracování práce, Ing. Jindřichu Dudkovi za praktické rady k práci a organizaci CESNET MetaCentrum VO za poskytnutí virtuálních zdrojů.

Obsah

Kapitola 1

Úvod

Autentizační protokoly se staly důležitou součástí všech dnešních systémů. Zajišťují zabezpečený přístup k počítačovým zdrojům, lze díky nim nastavit určitou míru restrikce přístupu v organizacích a velmi často jsou jediným způsobem zabezpečení velkých firemních síťových struktur, které obsahují důležitá a v některých případech velmi citlivá data. Například zájem o tyto data, ale také zájem na získání přístupu k nějakým systémovým strukturám, se často stává motivací pro snahu prolomit toto zabezpečení, čehož se nejlépe dosáhne obejitím nebo obelstěním těchto protokolů.

Hlavní náplní práce je analýza provozu v síťových strukturách, kde jsou nasazeny autentizační protokoly Kerberos, NTLM nebo SAML 2.0. Analýza probíhala především nad síťovým tokem, který byl ze sítě odchytáván v době, kdy probíhal útok na tyto protokoly. Útokem je myšlena situace, kdy se neautorizovaný uživatel pokoušel o přístup k datům a službám, ke kterým neměl dostatečné oprávnění, přičemž používal některé známé postupy, jak ochranu poskytnutou těmito protokoly obejít.

Způsoby, jak prolomit ochranu zabezpečovanou protokoly, jsou popsány v dalších kapitolách, neboť se různí dle každého protokolu. Různit se také mohou způsoby, jak útočit na síť z pohledu celé struktury, tedy jestli je útok proveden z vnitřku nebo vnějšku sítě, jestli má útočník přístup k administrátorským anebo pouze standardním účtům a tak dále. Pro každý protokol byly analyzovány různé scénáře, jejichž popis se nachází v dalších kapitolách.

Výstupem této analýzy měl být způsob, jak v co nejkratším čase a co nejjistěji identifikovat útočníka, který se snaží získat přístup do sítě, anebo již přístup do sítě získal a nyní se v ní pohybuje. Přístup, s jakým bylo k analýze a snaze o detekci útoků v této práci přistoupeno, byl kvůli cíli práce omezen pouze na pozorování síťové komunikace.

Hlavním cílem této práce bylo vytvoření pluginu ke Flowmon sondě, produktu firmy Flowmon Networks, volně dostupném na trhu, který měl získávat informace ze síťového toku tak, aby poté bylo možné automaticky detekovat útočníky, anebo alespoň upozornit na podezřelou aktivitu v síti.

Práce je rozdělena do šesti kapitol. První kapitolou je úvod a poslední závěr. V kapitole 2 se nachází popis základní teorie, která byla pro potřeby práce studována. V kapitole 3 jsou blíže popsány možné známe hrozby, kterým můžou protokoly zkoumané v této práci čelit. Kapitola 4 obsahuje popis praktické simulace útoků na protokoly, analýzy těchto protokolů v době takových útoků a následný návrh, jak řešit detekci těchto útoků. Následně kapitola 5 obsahuje popis implementace pluginu vytvářeného v této práci.

Kapitola 2

Teoretický základ

Svět počítačové komunikace a počítačových sítí může být pro mnohé jedním velkým zmatkem různých protokolů a standardů, které každý říkají něco jiného, a přesto nějak fungují. V této části práce jsou tedy sepsány nejdůležitější informace, jejichž znalost je potřebná při zabývám se danou problematikou. Postupně jsou zde popsány nejdříve síťový tok a způsoby jeho sledování, dále pak struktury, které mohou být zapojením strojů do sítě vytvářeny. Následuje popis zadaných síťových autentizačních protokolů, které jsou prozkoumávány více do hloubky, neboť je nutné kompletní porozumění zprávám, které tyto protokoly zasílají, aby mohla být jejich analýza co nejlepší, a v neposlední řadě také známé útoky na tyto protokoly a možnosti jejich detekce.

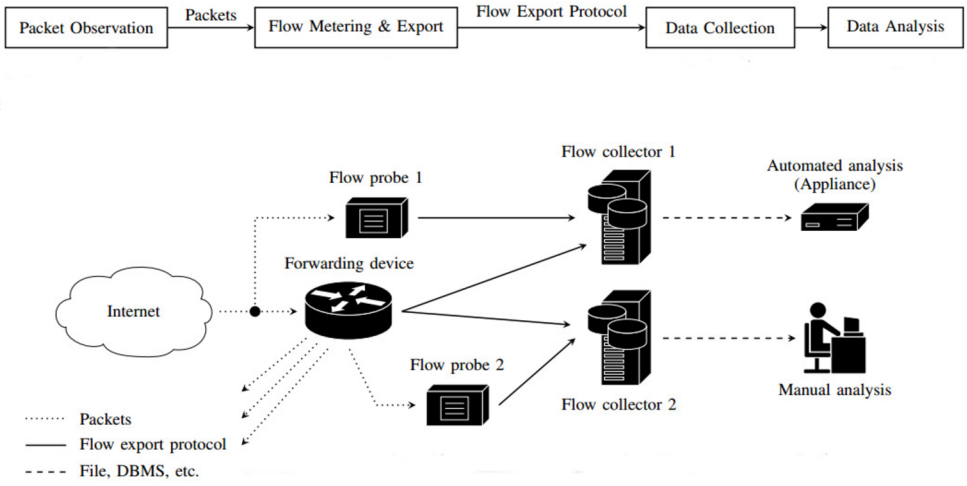
2.1 Síťový provoz a jeho monitorování

Definicí síťového toku (Net Flow) se v Internetové komunitě vyskytuje několik. Pro tuto práci byla používána definice dle standardu RFC 7011 [18], který se zabývá kontextem IPFIX (IP Flow Information eXport), kde je definován jako „Sbírka IP síťových paketů, které procházejí kolem určitého sledovaného bodu v síti po nějaký časový interval, mající takovou vlastnost, že všechny pakety náležící do určitého toku mají soubor společných vlastností“ [16].

Přístupy ke sledování síťového toku je možné rozdělit do dvou hlavních kategorií: aktivní a pasivní. Aktivní přístupy, pro něž se používají například nástroje jako `Ping` nebo `Traceroute`, samy naplní síť provozem, aby mohly provádět různé typy měření. Pasivní přístupy pozorují existující síťový provoz v určitém bodě sítě, tedy provoz vytvářený uživatelem. Jako příklad je možno uvést zachytávání síťových paketů, které obvykle poskytuje lepší pohled dovnitř síťového provozu, neboť umožňuje zachytávat a analyzovat celé síťové pakety. Tato práce se zabývá analýzou síťového provozu pomocí zachytávání a analýzy síťových paketů v provozu, tudíž je této metodě věnována větší pozornost. Názorné schéma sledování síťového toku je ukázané na obrázku 2.1 [16].

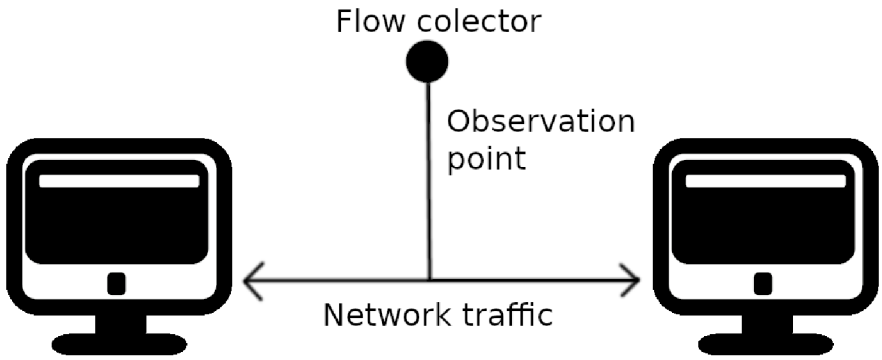
V souvislosti se síťovým tokem a jeho sledováním je třeba si upřesnit pár pojmů, které s touto problematikou velmi úzce souvisí. Jsou jimi:

- **Záznam o síťovém toku**, což je záznam obsahující informace o určitém síťovém toku, zachyceném na nějakém sledovacím bodě. Obsahuje naměřené vlastnosti toku (jako například celkovou velikost všech paketů v toku v bytech) a obvykle obsahuje charakteristické vlastnosti toku (jako například zdrojovou IP adresu) [16].



Obrázek 2.1: Schéma sledování a odchytávání síťového toku [16]

- **Bod sledování (Observation point)**, což je místo v síti, kde je možné pozorovat a odchytávat pakety. Každý bod je asociován se sledovací doménou (Observation domain). To je největší soubor sledovacích bodů, ze kterých lze poskládat informace o toku. Každý bod může být zároveň poskládán i z více bodů. Přibližné schéma popisující body sledování je na obrázku 2.2 [16].



Obrázek 2.2: Bod sledování(Observation point)

Proces sledování síťového toku

Proces sledování aktivity na síti a následného vytváření záznamu o této aktivitě, se většinou skládá z několika fází. Každá z těchto fází má své vlastní specifikace, které potom určují,

jak bude výsledný záznam vypadat. Při odchytávání je potřeba mít jasno v tom, za jakým účelem je monitoring prováděn, a jakým způsobem pak budou data využívána. Množství dat, které proudí přes síť, je většinou dost velké a pro případnou analýzu nevhodné. Je možné tedy v určitých fázích odchycená data upravit. Jednotlivé fáze jsou popsány níže.

Zachytávání síťových paketů

Jde o proces zachytávání a předzpracování síťových paketů. Balíčky musí být nejdříve zachyceny z síťové linky. Po zachycení dochází k několika kontrolám (například *checksum error check*), až poté je paket přesunut do paměti. Následuje označení časovou značkou (`timestamp`). Toto je důležité pro mnoho dalších procesů provádějících analýzu. Například pokud je potřeba kombinovat více paketů z různých míst odchytu do jednoho souboru, pakety budou vybírány podle jejich časové značky. Následuje oříznutí paketu, které je ovšem nepovinné a používá se pouze, aby se paket velikostí vešel do takzvané délky snímku (*snapshot length*), pokud je přednastavena. Toto pomáhá odlehčit náročnost následného zpracovávání a může být v důsledku žádané v případě, že jsou důležité hlavně hlavičky paketů, a ne jejich obsah. Na konci této fáze probíhá vzorkování a filtrace paketů. Vzorkování je motivováno redukcí dat vybráním podmnožiny nějakého souboru paketů tak, aby bylo stále možné určit všechny vlastnosti celého souboru. Filtrace je motivována odstraněním paketů, které nejsou pro monitorování zrovna zajímavé. Jak filtrace, tak vzorkování je nepovinné [16].

Vytváření a export síťového toku

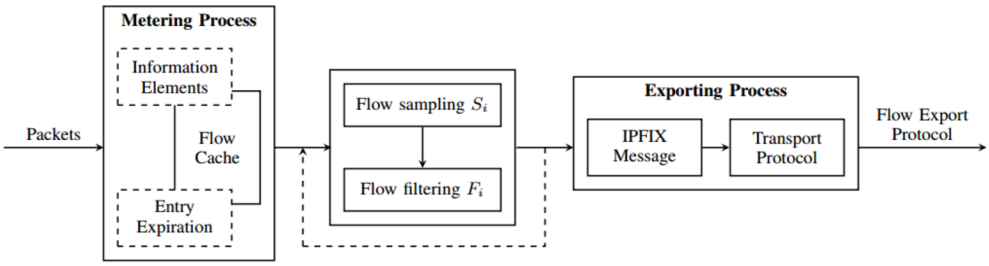
Fáze, u které jsou pakety seskupovány do toku a exportovány. Jde o klíčový prvek jakéhokoliv systému sledujícího síťový tok. Seskupování paketů do toku probíhá v *Procesu měření* podle informačních elementů, které jsou blíže popsány níže. Po seskupení jsou záznamy o tocích ukládány do tzv. *flow cache*, dokud není tok považován za skončený. Následuje volitelné vzorkování a filtrování toku. Poté záznamy o síťovém toku musí být zapouzdřeny ve zprávách. Také je nutné vybrat vhodný transportní protokol [16].

S exportem síťového toku úzce souvisí pojem **Informační elementy (IE)**. To jsou oblasti dat, které lze odeslat v záznamech o síťových tocích. Seznam základních IE udržuje IANA (Internet Assigned Numbers Authority). Elementy mají svoje jméno, číselné ID, popis, typ, délku a status. Za nejmenší množinu elementů popisujících síťový tok jsou považovány internetová adresa spolu s transportním protokolem, ale je možné je definovat pro jakoukoliv vrstvu síťové architektury [16].

Kolektivizace dat

Kolektivizace probíhá v kolektorech síťových toků. Jejich úkol je přijmout, uložit a předzpracovat data z jednoho nebo více exportních procesů v síti. V kolektorech běží jeden nebo více procesů, jejichž základní úkoly jsou komprese dat, seskupování dat, anonymizace dat, filtrování a vytváření souhrnu. Zajímavá je hlavně anonymizace, neboť data v zachyceném síťovém toku můžou představovat nebezpečí pro soukromí uživatele a můžou sloužit k jeho sledování. Nejjednodušší způsob je odstranění IP adres ze zachycených dat, ovšem to omezuje následné množství získatelných informací, a proto je vhodné pouze někdy. Jednou z nejlepších strategií je šifrovací algoritmus [16].

Na konci této fáze je záznam připraven pro analýzu, která se počítá jako finální fáze sledovacího procesu. Rozlišují se hlavní tři oblasti analýzy: Analýza síťového toku a hlášení, Detekce hrozeb a Monitorování výkonu. Pro účely této práce je více rozvíjeno detekování hrozeb [16]. Na obrázku 2.3 je možno vidět schéma celého výše popsaného procesu.



Obrázek 2.3: Schéma procesu odchytávání a exportu záznamu síťového toku [16]

Detekce hrozeb

Detekce hrozeb pomocí monitorování síťového provozu se dá přibližně rozlišit na dva druhy použití. První je pouhé sledování, kdo komunikoval s kým, potenciálně s informacemi o velikosti přenesených dat a počtu spojení. Druhý využívá definici síťového toku pro odhalování některých hrozeb, pro které jsou typická určitá chování sítě. Jako příklad se dá uvést detekce útoků jako DoS (Denial of Service) útok, skenování sítě apod. Jejich společným prvkem je, že ovlivňují metriku sítě, která se dá přímo získat ze záznamů o síťovém toku [16]. Dost často se však stává, že pouhým sledováním sítě není možné útok odhalit. Může to být díky povaze útoku, kdy se například útočník vydává za někoho jiného a po síti komunikuje pouze s touto falešnou identitou, nebo celé řadě jiných důvodů. V tuto chvíli může sledování síťového toku posloužit pouze jako varující element, který upozorní na podezřelou aktivitu. Potom však musí zafungovat jiný kontrolní prvek, nejlépe lidský, který toto upozornění zkontroluje a vyhodnotí [6].

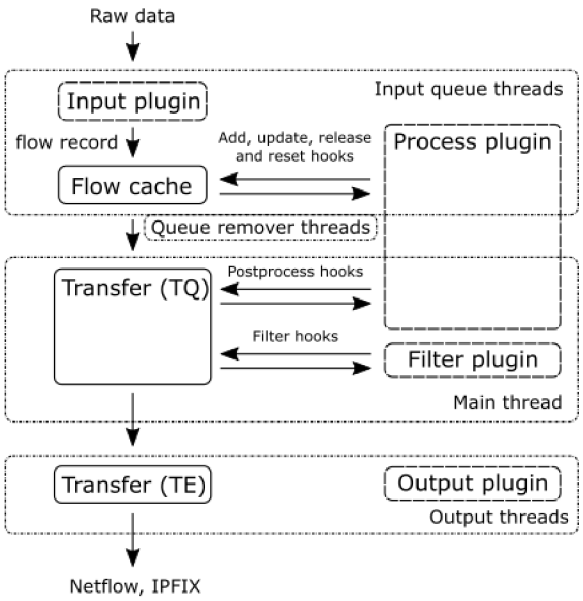
Nástroje pro sledování

Nástroje a aplikace pro sledování zpracovávají data o síťových tocích a ukazují sady metrik sítě pro každou sledovanou službu, což se používá k odhalení událostí na síti a jejich dopadu na koncového uživatele. Důležitým aspektem nástrojů pro sledování síťového provozu je jejich umístění v rámci sítě na místa, ze kterých je nejideálnější brát údaje o síťovém toku [16].

Cílem této práce bylo vytvořit plugin pro exportér jednoho konkrétního sledovacího nástroje, kterým je *Flowmon sonda*, dodávaná na trh firmou Flowmon.

Účelem exportéru, který je součástí Flowmon sondy, je transformovat provoz na síti na síťový tok. Poskytuje modulární architekturu s podporou různých formátů vstupních a výstupních dat, stejně jako komplexní zpracování dat. Může být spouštěn s různými pluginy, které definují jeho chování. V exportéru musí být vždy přítomný právě jeden z následujících druhů pluginů: vstupní, zpracovávající (procesní), filtrující a výstupní [9].

Vstupní plugin přijímá pakety ze sítě a poskytuje ostatním pluginům data ve formě kompletních záznamů o těchto paketech přes flow cache. Tyto záznamy jsou následně zpracovávány každým z procesích pluginů. Ty mohou záznam zpracovávat v době objevení no-



Obrázek 2.4: Struktura exportéru [9]

vého toku, objevení dalšího paketu náležícího do toku anebo po skončení toku. Po skončení zpracování toku je pomocí filtrujících pluginů rozhodnuto, jaké záznamy mají být exportovány a následně pomocí výstupních pluginů předány ve formě zpráv IPFIX, NetFlow nebo ve formátech jako json a csv [9]. Schéma fungování pluginu, jak je zde popsáno, je možno vidět na obrázku 2.4. Princi fungování exportéru a procesních pluginů je blíže popsán v kapitole 5.

Jako inspirace, použitá při nastudování, popisu a následné implementaci pluginu exportéru, může být uvedena bakalářská práce *Monitorovanie sieťových tokov protokolu Kerberos* [2] vypracovaná Matúšem Bačkorem na Fakultě Informatiky Masarykovy Univerzity, která se zabývá podobným tématem a jejím výstupem byl obdobný plugin zpracovávající zprávy protokolu Kerberos. V práci je bohužel pouze uvedeno, že plugin zprávu zpracovává, ovšem ne jak ji zpracovává.

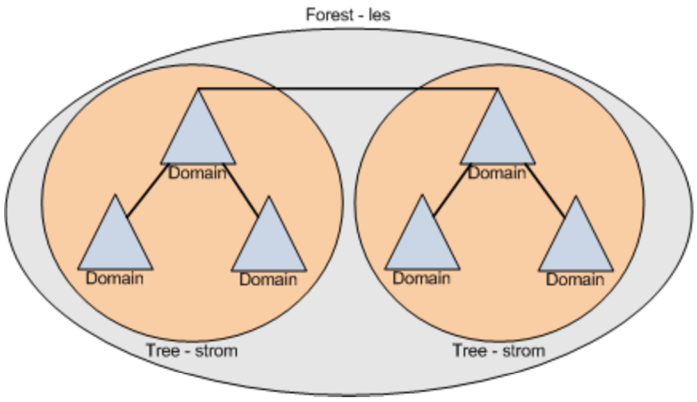
2.2 Síťové architektury a topologie

Důležitým aspektem, braným v potaz při analýze útoků na síť, je její architektura a topologie. Architektura, která je vystavěna tak, aby správně rozdělovala řízení do vrstev, a ve které jsou využívány moderní komunikační protokoly, které samy o sobě zajišťují bezpečný provoz, rozhodně bude vystavena menšímu riziku úspěšného útoku. Zároveň síť, jejíž logická topologie vzhledem k autentizačním postupům zajišťuje více vrstev a zajišťuje větší distribuci autentizačních autorit, je dalo by se říci zabezpečenější, neboť je více prostoru útočníka odhalit ve chvíli, kdy se v síti aktivně pohybuje. Možnosti útoku na sítě jsou popsány až dále.

V této práci, vzhledem ke zkoumaným protokolům, byly uvažovány především architektury a topologie používané ve firemní sféře, které jsou postavené na technologiích od firmy Microsoft, tedy například Active Directory. Tato, dá se říct, velmi úzká specifikace má jednoduché opodstatnění. Microsoft je momentálně nejrozšířenějším operačním systémem na světě [8], je hojně využíván jak pro osobní, tak pro firemní účely a všechny tři protokoly

jsou jeho službami ve velké míře podporovány [23, 25, 24], jak je dále popsáno v sekci 2.3. K tomu navíc je služba Active Directory, vzhledem k monopolu firmy Microsoft v oblasti operačních systémů, pravděpodobně první volbou adresářové služby, kterou mohou korporace používat pro svoje vnitřní sítě. Z toho vyplývá, že analýza provedená na útocích, které jsou prováděny v prostředí, které bude minimálně podobné mnohým dalším reálně používaným, může být využita pro mnoho dalších pokračování.

Každá firemní síťová struktura je unikátní, odpovídající firemním potřebám, tudíž ji nelze lehce generalizovat. Pokud ovšem je v síti využíváno Active Directory, musí být alespoň rámcově dodržena předdefinovaná struktura. Tato struktura je založena na hierarchickém rozdělení všech subjektů v síti, kdy na nejnižší úrovni figuruje klient, server anebo jiné zařízení (například tiskárna). Tyto síťové subjekty se sdružují do jednotlivých *organizačních jednotek* a následně *domén*, kde každá doména má vlastní *Domain Controller*. Domény se následně spojují do stromových struktur, které se nazývají *doménové stromy*. Více doménových stromů poté tvoří les. Alespoň rámcové schéma této struktury s Acitve Directory je viditelné na obrázku 2.5.



Obrázek 2.5: Obecné schéma síťové struktury s Active Directory [4]

Active Directory

Active Directory je rozšiřitelná adresářová služba obsažená v systému Windows, která umožňuje centralizovanou správu síťových prostředků. Umožňuje snadno přidávat, odebírat nebo přemisťovat účty pro uživatele, skupiny, počítače, stejně jako jiné typy prostředků. Služba je založena na standardních síťových protokolech [36].

Na tomto místě je nutné uvést definici důležitých pojmů souvisejících se službou Active Directory:

- **Adresář** je uložená kolekce informací o různých typech prostředků.
- **Adresářová služba** ukládá veškeré informace potřebné k použití a správě distribuovaných prostředků na jednom místě, umožňuje spolupráci těchto prostředků a je zodpovědná za ověření přístupu, správu identit a kontrolu vztahů mezi prostředky.

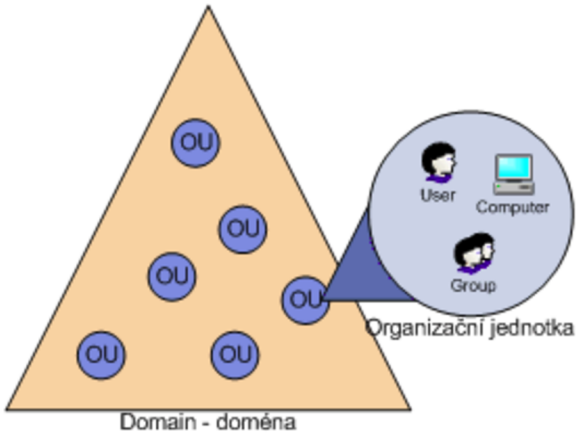
Active Directory je navržena, aby vzájemně spolupracovala s dalšími adresářovými službami, a aby akceptovala požadavky od různých klientů využívající různá rozhraní. Primárním protokolem je LDAP (Lightweight Directory Access Protocol), pro práci s jinými ser-

very je používán REPL (read-eval-print loop). Služba Active Direcory je velmi úzce svázána se službou DNS (Domain Name Service) [36].

Logická struktura Active Directory

Služba Active Directory v sobě obsahuje nadefinované základní logické součásti. Jsou jimi doména, strom domén, doménové struktury a organizační jednotky.

Doména je logické seskupení objektů, které sdílí společné databáze pro službu Active Directory; je možné tam vytvářet účty pro uživatele, skupiny, počítače a sdílené prostředky. Domény se sdružují do stromů a z nich vznikají doménové struktury [4, 22]. Schéma domény je možné vidět na obrázku 2.6.



Obrázek 2.6: Vnitřní schéma domény, příklad organizační jednotky [4]

2.3 Síťové autentizační protokoly

Síťové autentizační protokoly jsou takové protokoly, které jsou schopné ověřit identitu nějakého subjektu v síti [7], a tím zajišťují kontrolu omezení přístupu k určitým datům nebo službám v síti. Tato restrikce je založena na přidělení jistého množství práv jednotlivým uživatelům nebo systémům, které na základě těchto práv mají, anebo nemají možnost k službám nebo datům přistupovat. Protokolům se uživatelé nebo systémy prokazují pomocí své identity, která je většinou založena na kombinaci jejich uživatelského jména a hesla. Útočník, který chce proniknout do sítě ke službám nebo datům, ke kterým nemá přístup, může takto učinit například pomocí odcizení této identity [1].

V následující části jsou blíže popsány protokoly Kerberos, NTLM a SAML 2.0, přičemž důraz je kladen především na to, jakým způsobem protokoly realizují komunikaci klient-server.

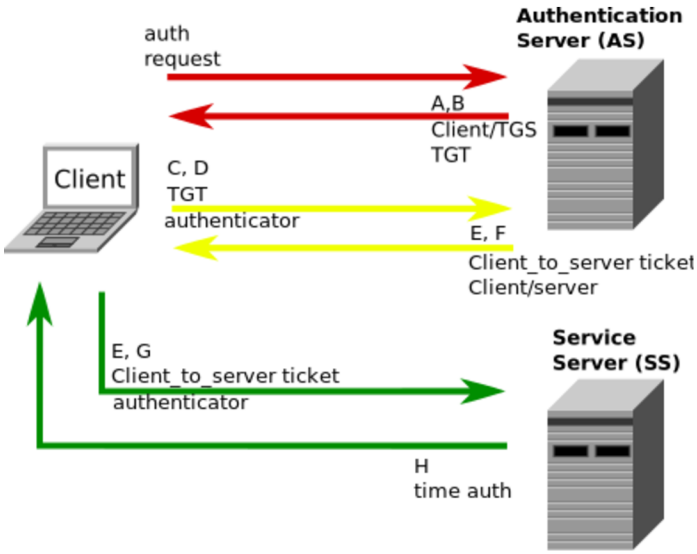
Kerberos

Kerberos je autentizační protokol vytvořený MIT (Massachusetts Institute of Technology) jako 'řešení problémů se zabezpečením sítí'. Používá silné šifrování, tudíž se klient může identifikovat na server i skrz nezabezpečené připojení. Dále také může klient i server zašifrovat veškerou svou další komunikaci. Kerberos je považován za důvěrný protokol třetí

strany, což znamená, že každý klient důvěřuje protokolu Kerberos, co se týče věrohodnosti identity všech ostatních klientů [28].

Kerberos používá pro autentizaci databázi všech klientů a jejich *privátních klíčů*. Privátní klíč je velké číslo známé pouze protokolu Kerberos a klientovi, kterému náleží (v případě, že je klientem uživatel, jde o zašifrované heslo) a je ustanoveno při registraci klienta do databáze. Tento klíč je vypočítáván pomocí kryptografického algoritmu AES, s reprezentací klíče na 128 nebo 256 bitech [14]. Tímto Kerberos ví o všech klientech a může dokazovat totožnost. Zároveň je používán i jiný druh klíčů (takzvané *session keys*), které můžou být poskytnuty dvěma klientům a slouží k šifrování komunikace mezi nimi [37].

Autentizace v Kerberosu je prováděna pomocí zvláštních zpráv, kterými si uživatel žádá o přístup k službám a pomocí kterých se prokazuje, že je oprávněn službu použít [37]. Základní schéma fungování protokolu Kerberos je možno vidět na obrázku 2.7.



Obrázek 2.7: Schéma komunikace protokolu Kerberos [42]

Proces autentizace

V autentizačním modelu figurují dva typy přístupových práv: *tickets* – tikety a *authenticators*. Tiket je používán k bezpečnému předání identity klienta, kterému byl povolen přístup k nějaké službě. Též obsahuje informaci použitelnou ke kontrole, že klient, který tiket použil, je ten komu byl určen, tudíž nejde o ukradený tiket. Authenticator obsahuje informace o klientovi, který tiket zaslal a slouží ke srovnání totožnosti popsané výše [37].

Jeden tiket je použitelný pro jednoho klienta a jeden server, je šifrovaný pomocí klíče serveru, kde má být uplatněný a je použitelný k opakovanému použití, dokud nevyprší jeho platnost. Struktura je následující:

$$\{s\_name, c\_name, c\_addr, timestamp, tckt\_life, sess\_key\}$$

Kde:

- `s_name` je jméno serveru,
- `c_name` je jméno klienta,

- `c_addr` je síťová adresa klienta,
- `timestamp` je čas vystavení tiketu,
- `tckt_life` je čas po který je tiket platný,
- `sess_key` je náhodný session key pro dvojici klient a server.

Authenticator je naopak použitelný pouze jednou, tudíž musí být vytvářen znovu pro každou žádost o autentizaci. Toto nepředstavuje problém, protože obsahuje informace, které může dodat sám klient. Struktura je:

$$\{c\_name, c\_addr, timestamp, sess\_key\}$$

Uživatel má na začátku k dispozici pouze jeho heslo, kterým může prokázat svoji totožnost. Autentizace probíhá následovně:

1. Uživatel zadá své uživatelské jméno a heslo. Jméno je odesláno autentizačnímu serveru spolu s názvem speciální služby, která se nazývá *ticket-granting service* (TGS).
2. Pokud autentizační server má záznam o klientovi ve své databázi, vygeneruje session key pro pozdější komunikaci klienta a takzvaného *ticket-granting* serveru, a obsáhne ho, spolu s jménem a adresou klienta, v tiketu, které odešle ticket-granting serveru, a který je zašifrován klíčem známým pouze těmto dvoum serverům.
3. Stejný tiket je odeslán i klientovi, je ovšem zašifrován pomocí privátního klíče klienta.
4. Na straně klienta je po přijetí tiketu převedeno uživatelovo heslo na kryptografický klíč a použito pro dešifrování tiketu, který se následně uloží. Tento tiket se také nazývá Ticket Granting Ticket (TGT).

V tuto chvíli má klient dost informací, aby se dokázal autentizovat jakékoliv službě. Tento proces probíhá následovně:

1. Pokud klient nevlastní tiket pro prokázání se nějakému serveru, tak o něj zažádá dotazem na ticket-granting server, který obsahuje jméno serveru, pro který je žádán přístup, tiket popsaný výše a authenticator.
2. Ticket-granting server po přijetí zkontroluje platnost dotazu (pomocí authenticatoru), pokud je v pořádku, tak vytvoří nový session key pro komunikaci klienta a serveru, pro který se tiket vystavuje, vytvoří tiket (nazývá se client-server ticket), zašifruje ho pomocí session key přijatého od autentizačního serveru a odešle klientovi, který po přijetí tiketu od ticket-granting serveru již může prokázat svoji totožnost žádanému serveru.
3. Klient sestaví authenticator a zašifruje ho pomocí session key přijatého v tiketu od ticket-granting serveru. Ten potom i s ticketem odešle na žádaný server.
4. Server po přijetí dešifruje authenticator, porovná informace v něm obsažené s těmi v tiketu a porovná čas s aktuálním časem (pokud se výrazně liší, server bere dotaz jako pokus o znovupoužití již dříve použitého dotazu).
5. Pokud všechny předchozí kroky proběhly v pořádku, tak klient dle protokolu Kerberos úspěšně prokázal svoji totožnost a může mu být připuštěn přístup ke službě, navíc klient a server oba mají klíč, který je známý jen jim dvěma, tudíž si můžou být jistí, že zpráva obsahující tento klíč, je jistě od jejich protějšku [37].

NTLM

NTLM (NT[1] LAN (Local Area Network) Manager) je autentizační protokol využívaný v sítích, které zahrnují stroje s operačním systémem Windows a na samostatně jedoucích systémech. Postupně je nahrazován lepším a bezpečnějším protokolem Kerberos, ovšem je stále podporován a, především na starších systémech, stále používán [24].

Přístupová práva udělovaná protokolem jsou založená na datech získaných od uživatele, tedy na jménu domény, uživatelském jménu a zahashovaném heslu (one-way hash). NTLM využívá pro autentizaci šifrovaný protokol, nicméně bez toho, aby někam posílal uživatelovo heslo [24].

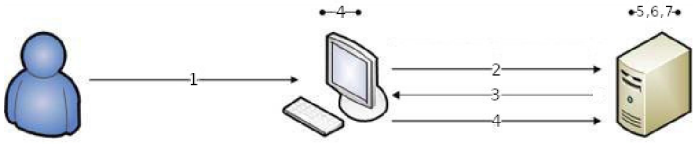
Pro autentizaci v rámci sítě jsou potřeba minimálně dva systémy: klient, kde uživatel vyžaduje přihlášení a takzvaný *domain controller*, kde jsou uložena data spjata s uživatelovým heslem [24].

Proces autentizace

Autentizace pomocí NTLM probíhá následujícím způsobem:

1. Uživatel zadá klientskému systému jméno domény, uživatelské jméno a heslo, systém vypočítá z hesla jeho hash hodnotu a původní heslo zahodí.
2. Klient zašle uživatelské jméno serveru (ve formátu `plaintext`).
3. Server vygeneruje 16-ti bitové číslo které se nazývá *challenge*, a odešle ho klientovi.
4. Klient zašifruje tuto challenge pomocí hashe získaného z hesla a výsledek odešle serveru (*response*).
5. Server odešle uživatelské jméno, challenge a response na domain controller.
6. Domain controller pomocí uživatelského jména zažádá o hash jeho hesla v databázi, které použije na zašifrování challenge.
7. Proběhně porovnání hodnoty hodnoty challenge zašifrované klientem a controllerem, pokud jsou stejné, tak autentizace proběhla úspěšně.

Ve chvíli, kdy je server, vůči kterému probíhá autentizace, zároveň i domain controller, odpadá komunikace v bodě 5. Schéma autentizace je možno vidět i na obrázku 2.8.



Obrázek 2.8: Schéma autentizace NTLM [35]

V dnešní době jsou systémy Windows nastaveny tak, aby používaly Kerberos, pokud je to možné. Obecně je protokol NTLM považován za zastaralý a nebezpečný, protože používá slabou metodu šifrování [24, 26, 27].

[1] Označení řady operačních systémů firmy Microsoft

SAML 2.0

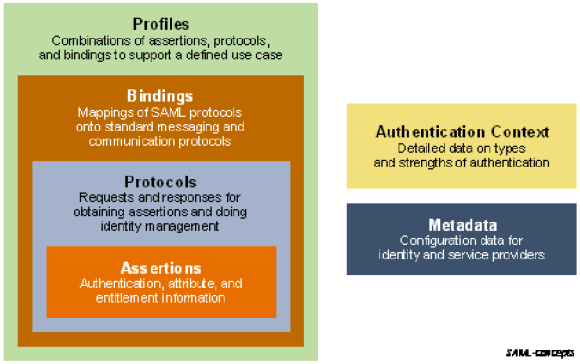
SAML (Security Assertion Markup Language) 2.0 je verze frameworku pro výměnu bezpečnostních informací mezi partnery v rámci sítě. Je vyvíjen organizací OASIS (the Organization for the Advancement of Structured Information Standards) [30].

SAML je framework založený na jazyku XML. Hlavní informace, které přenáší, jsou tvrzení (assertions) s jasně danou syntaxi a pravidly, které potvrzují důvěru síťových aplikací i za hranicemi jejich domény. Zjednodušeně popsáno, je používán k zasílání zpráv, potvrzujících totožnost již přihlášeného klienta, mezi službami, které samostatně vyžadují autentizaci, a tudíž nezatěžuje klienta s opakovanou autentizací [30].

Hlavními principy, které vedly k potřebě vytvoření takového nástroje jsou například:

- **Single Sign-On (SSO)** – princip jediného přihlášení k webový doménám, které bude platné pro více domén bez přispění uživatele,
- **Společná/sdílená identita (Federated identity)** – identita uživatele sdílená skrz více systémů a přijatá všemi partnery v určité skupině, úzce spjaté se SSO [30].

Koncept SAMLu je složen ze blokových komponentů, které díky své modularitě umožňují pokrytí mnoha případů užití. Základní komponenty, u kterých je jasné definovaná struktura a obsah, jsou **tvrzení** (assertions, obsahují stanoviska, které strana, od které pochází, prohlašuje za pravdivé) a **protokoly**(zprávy pro zasílání a přijímání tvrzení a zprávu identity). Další jsou **návaznosti** na standardní běžně užívané komunikační protokoly (například HTTP – Hypertext transfer protocol) a **profily**, které jsou přesně navržené aby kombinovaly předchozí tři komponenty a pokryly požadavky dané služby [30]. Blokové schéma protokolu SAML je možné vidět na obrázku 2.9.



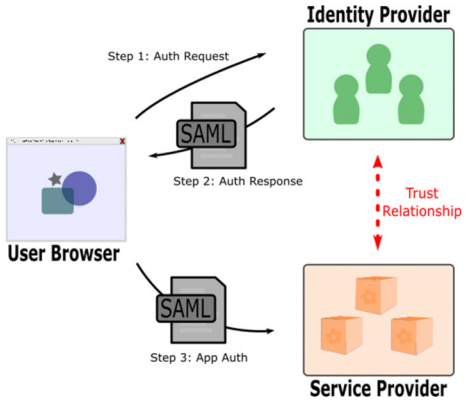
Obrázek 2.9: Blokové schéma SAML 2.0 [30]

Obecně je SAML využíván nejvíce pro službu SSO. V prostředí Active Directory je to především při integraci aplikací třetích stran. V interakcích, které probíhají pomocí protokolu SAML, figurují vždy minimálně dvě entity. Ty jsou nazývány SAML *asserting party* nebo *authority* a SAML *relying party*. SAML authority je systém, který vystavuje tvrzení, relying party poté systém, který tyto tvrzení využívá.

Princip SSO je následující. Je předpoklad, že uživatel získal přístup k nějakému systému, například webové stránce, pomocí prokázání své identity a následně, například kvůli přesměrování, zažádal o přístup k jinému systému, který je s prvním systémem partnerský a je mezi nimi sdílena uživatelova identita. V tuto chvíli proběhne mezi těmito dvěma systémy interakce, kdy první systém poskytuje tvrzení druhému systému, že identita uživatele

je již známa. Vzhledem k partnerskému vztahu těchto dvou systémů, druhý systém věří tvrzení prvního, a tedy poskytne uživatelovi přístup ke svým zdrojům. Schéma demonstrující použití SSO pomocí protokolu SAML je možno vidět na obrázku 2.10.

Existují aplikace, které je možné napojit na Active Directory, a tudíž nastavit, aby pro ověření přístupu pro SSO využívaly jako autoritu pro ověření uživatelovy identity autentizační systém v Active Directory [30, 31].



Obrázek 2.10: SSO pomocí SAML 2.0 [20]

Protože protokol SAML je založen na zprávách založených na syntaxi XML, je velké množství způsobů, jak může být zpráva koncipována. Zde na obrázku 2.11 je přiložen příklad zprávy SAML assertion s jednoduchým autentizačním prohlášením (část *Assertions* na obrázku 2.9).

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  Version="2.0"
  IssueInstant="2005-01-31T12:00:00Z">
  <saml:Issuer Format=urn:oasis:names:SAML:2.0:nameid-format:entity>
    http://www.example.com
  </saml:Issuer>
  <saml:Subject>
    <saml:NameID
      Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
        j.doe@example.com
    </saml:NameID>
  </saml:Subject>
  <saml:Conditions
    NotBefore="2005-01-31T12:00:00Z"
    NotOnOrAfter="2005-01-31T12:10:00Z">
  </saml:Conditions>
  <saml:AuthnStatement
    AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="67775277772">
    <saml:AuthnContext>
      <saml:AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
      </saml:AuthnContextClassRef>
    </saml:AuthnContext>
  </saml:AuthnStatement>
</saml:Assertion>
```

Obrázek 2.11: Příklad zprávy SAML [30]

Kapitola 3

Zhodnocení hrozeb pro protokoly

Každý autentizační protokol, ač sebelépe vymyšlený, je napadnutelný. Způsobů, jak napadnout nebo obejít bezpečností protokoly, je velmi mnoho [1]. Případné útoky ještě ulehčuje, když je protokol implementován v rámci sítě, tudíž komunikace probíhá přes většinou nezabezpečený kanál, kde může kdokoliv odposlouchávat komunikaci mezi ověřujícím serverem a autentizujícím se klientem a následně použít získané informace k prolomení ochrany poskytované protokolem. Cílem této práce je sledovat přesně ty případy, kdy jde o útok na síťovou strukturu, kde jsou implementovány zkoumané protokoly.

Pokrýt všechny druhy útoků, které jsou zdokumentované pro dané protokoly, by bylo v rámci jedné práce skoro nemožné. Proto je pro každý protokol popsán právě jeden útok. Každý tento útok byl vybrán kvůli jeho popularitě, a tedy i případné další využitelnosti jeho analýzy. V pořadí, v jakém byly v předchozí kapitole popsány protokoly, jsou v této kapitole popsány jim odpovídající útoky. Zároveň je popis doplněn také o návrhy, jak by bylo možné tyto útoky v rámci sítě detekovat, a tedy také s jakou myšlenkou bylo prováděno prvotní testování. Také je zde stručně popsáno, jak by mohl probíhat útok, který by měl za cíl postupně ovládnout celou síť.

3.1 Útok na protokol Kerberos – Pass the Ticket

Vzhledem ke komplexnosti protokolu Kerberos je mnoho způsobů, jak protokol napadnout. Pro úplnost jsou zde uvedeny ty nejzajímavější, kterými se ovšem tato práce nezabývá:

- **Prolomení hrubou silou**, je způsob, jakým je možné v kryptografii prolomit jakoukoliv šifru, anebo v oblasti počítačové uhádnout jakékoliv uživatelské heslo. Vzhledem k tomu, že se uživatel vůči protokolu autentizuje pomocí kombinace uživatelského jména a hesla, je tento způsob možné využít při útoku na protokol Kerberos. Z pohledu analýzy síťového provozu však nejde o tak zajímavý útok, neboť znamená markantní zvýšení provozu na síti, které je lehce detekovatelné [33, 3].

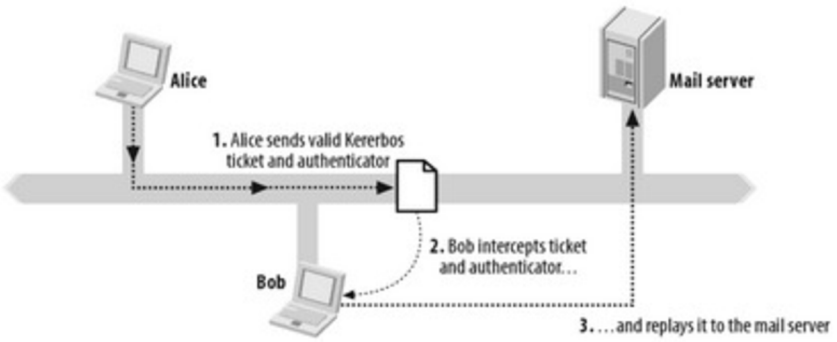
- **Kerberoasting** je již trošku zajímavější druh útoku. Jeho principem je odchytávání tiketů ze síťového provozu, které jsou šifrovány pomocí klíčů získaných z uživatelských hesel. Tím pádem lze z těchto tiketů získat hash hodnotu uživatelského hesla, které poté může být využito [33]. V tu chvíli již jde o jiný útok s názvem **Pass the Key** nebo **Pass the Hash**, který je blíže popsán v 3.2.

Protokol Kerberos se může stát terčem i jiných druhů útoku (Replay attack, Man-in-the-middle attack [10]), ovšem proti těmto bývá většinou chráněn ze své podstaty způsobem, jakým je implementován.

Útok **Pass the Ticket** je ikonickým útokem na protokol Kerberos. Při tomto útoku je, podobně jako při útoku Kerberoasting, získán tiket. V tomto případě je ovšem přímo tento tiket použit k autentizaci útočníka [33].

Odchytit, a tedy ukrást, je možno buď klient-server tikety, anebo TGT tikety. V případě, že je tiket úspěšně ukraden, útočník si ho uloží do svého systému a následně se jeho pomocí může autentizovat. V případě, že byl ukraden klient-server tiket, má útočník možnost přístupu pouze k jedné službě anebo stroji. Ovšem v případě, kdy se podaří útočníkovi ukrást TGT tiket, může díky němu získat přístup kamkoliv, kam má přístup uživatel, kterému tiket originálně náleží. Použitelnost tiketu je samozřejmě omezena jeho platností, případně dobou, po kterou může být obnoven [40, 33].

Na obrázku 3.1 je vidět schéma, jak by mohl vypadat útok Pass the Ticket.



Obrázek 3.1: Schéma útoku Pass the Ticket [10]

Další možnosti útoku

Použitelnost každého ukradeného tiketu úměrně roste s množstvím přístupových práv, které náleží původnímu vlastníkovi. Pokud se podaří získat tiket řadového uživatele, který má přístup pouze k jednomu vzdálenému systému, není to pro útočníka tak výhodné, jako když ukradne tiket síťového super-administrátora, který má neomezený přístup úplně ke všem službám a datům v celé síti. Nicméně i s pomocí úplně základního tiketu může útočník provést velmi velký a rozsáhlý útok.

V této práci není důležitý, a tudíž není blíže rozváděn, způsob, jakým útočník získá prvotní přístup k síti. V případě protokolu Kerberos tedy není důležité, jak útočník získá první tiket. Popis, jak byl získáván tiket v rámci této práce ve fázi praktické analýzy útoků, je v sekci 4.2. Důležité je to, co následuje.

Platí předpoklad, že útočník získal vlastnictví tiketu uživatele, který má přístup právě k jednomu vzdálenému systému. Pomocí tohoto tiketu se autentizuje vůči tomuto systému, jak bylo popsáno v sekci 2.3. Nyní tedy má přístup ke službám a datům, dostupným na tomto systému. Zároveň ovšem může využít skutečnosti, že protokol Kerberos je nastaven tak, že po uložení tiketu do nějakého systému, je zde tento tiket zachován po celou dobu, kdy trvá sezení daného uživatele [29] a někdy dokonce i po jeho skončení (vlastní zkušenost). Nejspíše je to z důvodu možnosti obnovit tiket po nějakou dobu, nastavenou protokolem. Pokud jsou tedy v systému uloženy tikety jiných uživatelů, útočník je může

extrahovat. Pokud nejsou, má také možnost použít napadený systém jako místo, ze kterého bude odchytávat tikety putující po síti. Opakováním tohoto postupu se dříve nebo později útočník nejspíše dostane k tiketu síťového super-administrátora, díky kterému získá neomezený přístup k celé síťové struktuře.

Jiným způsobem, jak provést útok, jehož důsledkem je ovládnutí celé síťové struktury, ale který nebyl v této práci zkoumán a opět je uveden pouze pro úplnost, je kombinace takzvaných **Silver ticket** a **Golden ticket** útoků. Jde o způsob vytvoření vlastního tiketu (klient-server tiket pro Silver ticket, TGT pro Golden ticket), což není vůbec těžké, neboť struktura tiketů odpovídá standardům, do kterého jsou zapsány maximální přístupová práva a je zakódován pomocí hashe uživatelského hesla (popsáno v sekci 2.3), které je nutné předtím získat, což je blíže přiblíženo v sekci 3.2. Pro útok Golden ticket je ovšem potřeba získat hash z účtu *krbtgt* [33]. Důvod, proč nebyly útoky zkoumány je, že z pohledu síťové komunikace se neliší od útoku Pass the Ticket.

3.2 Útok na protokol NTLM – Pass the Hash

Jak již bylo zmíněno výše, protokol NTLM je zastaralý a kvůli slabé metodě šifrování v systémech Windows potlačován. Nicméně stále se využívá jak pro lokální přihlášení, tak pro některé jiné služby v Active Directory, například webové [24].

Největší slabost protokolu NTLM spočívá, díky výkonnosti dnešních strojů, v celkem rychlém a jednoduchému prolomení hrubou silou (útok popsán v sekci 3.1). Opět však tento útok může být v rámci sítě lehce detekovatelný, kvůli rychlém nárůstu provozu v síti. Zajímavější a rozhodně podstatně hůře detekovatelný je tedy útok Pass the Hash.

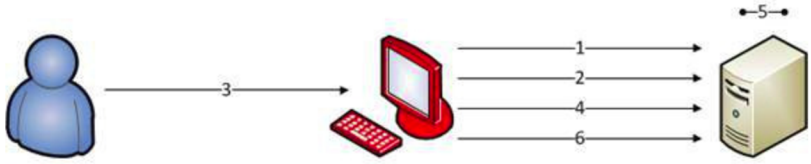
Útok **Pass the Hash** využívá způsobu, jakým jsou ukládána uživatelská hesla v systémech. V sekci 2.3 je popsán způsob, jakým kontroluje protokol správnost uživatelského hesla. Hlavní myšlenkou tohoto útoku, je namísto uživatelského hesla poskytnout systému přímo hash hodnotu tohoto hesla. Protože přepočet hesla na jeho hash hodnotu se provádí na straně uživatele, nejde o narušení postupu, kterým se protokol řídí. Útočník se tedy vůči systému autentizuje kombinací hodnot jména a hash hesla, čímž pádem získá stejný přístup, jako by poskytl uživatelské heslo. Průběh tohoto útoku může být následující:

1. Uživatel zažádá o přístup ke službě.
2. Server zašle autentizační `challenge`.
3. Uživatel zadá jméno a ukradenou hash hodnotu hesla.
4. Hash hodnota je zaslána na server.
5. Na serveru proběhne porovnání přijaté hodnoty oproti očekávané hodnotě.
6. Uživatel získá přístup ke službě.

Schéma tohoto útoku je možné vidět na obrázku 3.2. Opět, způsob získání hodnoty hash není předmětem zkoumání v této práci, ovšem pro úplnost je popsán dále v této sekci [41, 35].

Další možnosti útoku

Podobně jako si u předchozího útoku Pass the Ticket systém ukládá přijaté tikety, tak je hash hodnota hesla ukládána do speciálního SAM (The Sequence Alignment/Map) souboru.



Obrázek 3.2: Schéma útoku Pass the Hash [35]

Tento soubor není volně přístupný, pokud je systém běžící, ovšem jsou způsoby, jak k souboru přistoupit jinak [19]. Útočník může po získání přístupu k systému tento soubor otevřít a z něho získat hash hodnoty uživatelů, kteří se k systému do té doby přihlásili. Postupem podobným, jako byl popsán v sekci 3.1, může útočník postupnou prací získat hash hesla super-administrátora a tím získat přístup k jeho účtu, a tedy k celé síťové architektuře.

Útočník může také kombinovat útoky Pass the Hash a Pass the Ticket. Jako příklad takového případu může být uvedena třeba situace, kdy získal útočník pomocí ukradeného tiketu přístup k nějakému systému a chtěl by zde provést extrahování uložených tiketů. Při tomto příkladu, stejně jako ostatně v celé práci, je uvažován systém Windows, který má nastaveny omezení, které způsobují, že přehled o všech tiketech uložených v systému má přehled pouze administrátorský účet. Útočník může tedy extrahovat hash administrátorského hesla pro tento systém, pomocí něho získat přístup k administrátorským právům a následně extrahovat tikety, které následně může opět použít jinde.

Toto je pouze jeden z mnoha možných scénářů napadení sítě a je zde uveden opět pouze pro ilustraci, protože čistě z pohledu sledování síťového provozu by bylo velmi náročné takovýto útok sledovat, a proto se jím tato práce nezabývala. Nicméně rozhodně může být tímto směrem prováděn další výzkum.

3.3 Zneužití zprávy protokolu SAML

Předchozí dva protokoly fungují na podobných principech. V systémech většinou stojí vedle sebe (je možné zvolit si jeden anebo druhý pro stejnou věc), a tudíž mají i spoustu podobných rysů a i přístup k nim je dost podobný. SAML na druhou stranu je protokol, jehož použitelnost začíná tam, kde často končí použitelnost dvou dříve zmíněných protokolů. Pomocí SAMLu se většinou neřeší prvotní autentizace uživatele, ale bývá využíván až při prokazování identity již autentizovaného uživatele třetím stranám. Tímto se mimo jiné velmi zvyšuje množství způsobů, jak může být protokol využíván, a tedy i množství různých útoků, které můžou být na protokol provedeny.

Prokázání identity uživatele pomocí protokolu SAML probíhá pomocí zpráv. Tyto zprávy jsou však zasílány přes uživatelův prohlížeč, kde s nimi může být jednoduše manipulováno. Nejdůležitější částí zprávy protokolu SAML je část *Assertions*. Ta obsahuje prohlášení strany, která poskytuje tvrzení o věrohodnosti uživatele, který toto žádá, a podpis této strany, dokazující její identitu a tedy důvěryhodnost [30]. Útočník může tedy tuto putující zprávu odchytit a upravit tak, aby mu posloužila k autentizaci vůči nějaké službě bez nutnosti poskytnutí autentizačních údajů.

SAML sám o sobě nemá žádné mechanismy, aby takovému zneužití zabránil. Každá aplikace, implementující například službu SSO pomocí protokolu SAML, může obsahovat vlastní implementaci s vlastní sadou chyb. Dále jsou uvedeny ty nejčastější.

- Chybějící údaj o expiraci. Pokud zprávy protokolu SAML neobsahují údaj o vystavení a expiraci, můžou být jednoduše znovu použity kdykoliv v budoucnu.
- Nejedinečnost zpráv. Pokud zprávy neobsahují údaj, který zajišťuje jejich jedinečnost, a není nastavena kontrola této jedinečnosti, můžou být stejné zprávy používány stále dokola.
- Nepřítomnost nebo neplatnost podpisu. Pokud zpráva neobsahuje podpis strany vydávající tvrzení, je velice jednoduché zprávy falšovat. Navíc, pokud přijímající strana nemá nastavenu důslednou kontrolu, může být podpis přítomen, ale zfalšován.
- XML External Entity. Zpráva SAML je pouze druh zprávy XML, tudíž na ní můžou být použity útoky jako na XML [20].

Obecně existuje možných druhů útoků na protokol SAML velice mnoho a nelze je lehce unifikovat.

Je rozhodně zajímavé ovšem zmínit útok, který se v mnohém podobá již zmiňovanému útoku Golden ticket. Jde o útok s názvem **Golden SAML** a s jeho použitím je možné získat přístup k jakémukoliv systému využívajícího SAML s jakýmikoliv právy. Jeho princip spočívá ve vygenerování zprávy protokolu SAML nezávisle na tom, na jakém systému se útočník nachází a pouze za předpokladu, že útočník má přístup k uživatelskému účtu v doméně a k určitým informacím. Ty jsou následující:

- privátní klíč pro podpis,
- veřejný certifikát patřící autoritě ověřující identitu,
- jméno autority ověřující identitu,
- role v systému (např. admin).

Všechny tyto informace je velice jednoduché získat za předpokladu, že má útočník přístup k uživatelskému účtu. Následně je s jejich pomocí vygenerována SAML assertion, která je s pomocí privátního klíče podepsána. V tuto chvíli má útočník k dispozici zprávu SAML, kterou může bez problémů použít k autentizaci.

Tento útok obecně není považován za hrozbu, především z důvodu nutnosti přístupu k uživatelskému účtu, ovšem za použití postupů popsaných u předchozích dvou protokolů je i tento přístup možné získat [34].

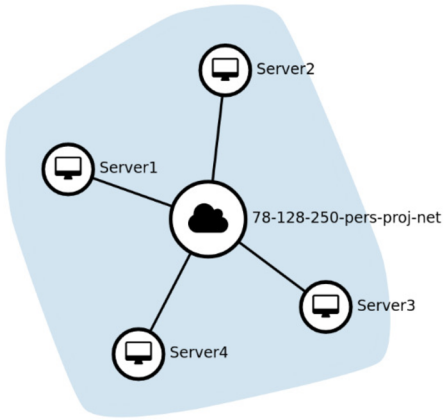
Kapitola 4

Analýza protokolů a návrh detekce útoků

Tuto kapitolu je možno rozdělit do dvou jednotlivých celků, které spolu ovšem úzce souvisí. V předchozích kapitolách bylo specifikováno, jakým způsobem bylo v práci nahlíženo na problém analýzy bezpečnostních protokolů. Tento problém byl řešen z pohledu sledování síťového provozu v síťové struktuře, která má být podobná té, jakou můžeme najít například ve vnitřních sítích firem anebo korporací, a která je postavena na technologiích a principech Microsoft Windows a Active Directory.

Podobná síť byla uměle vytvořena, byla v ní nastavena autentizace pomocí zkoumaných protokolů a byl odchytáván její vnitřní provoz v době, kdy v síti probíhaly simulované útoky. Popis této činnosti, včetně popisu provedení útoků, můžeme chápat jako první ze zmíněných celků. Druhým pak je popis zkoumání záznamů z tohoto provozu a následného návrhu, jakým způsobem nejlépe detekovat prováděné útoky.

V této a všech následujících kapitolách již není zmiňován protokol SAML. Je to z toho důvodu, že SAML není využíván v systémech Windows v základní výbavě, tudíž je potřeba uměle přidat autentizaci s jeho pomocí. Bohužel, aplikace využívající protokol SAML jsou většinou určeny velkým korporací a všechny jsou placené, a tudíž pro akademické účely nevhodné. Protokol SAML tedy nebyl v této práci zkoumán z praktického hlediska. Další postup, který by mohl být zvolen při zkoumání tohoto protokolu je diskutován v závěru.



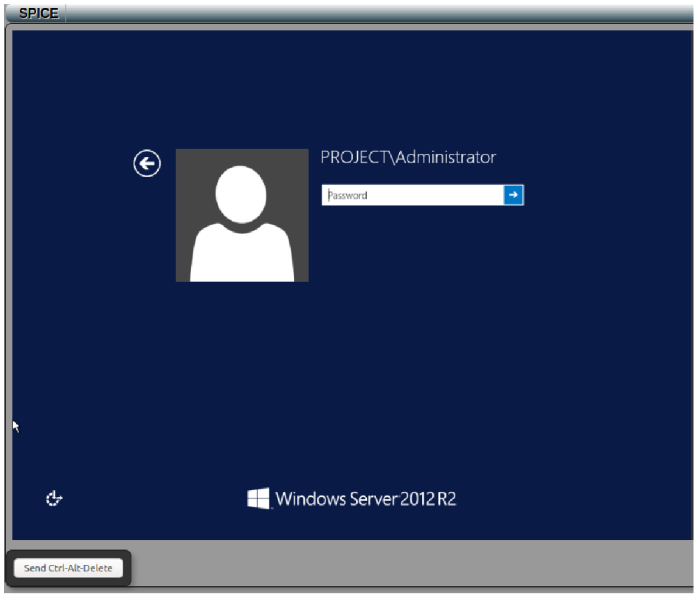
Obrázek 4.1: Schéma vytvořené sítě

4.1 Prostředí pro provádění analýzy

Pro provádění analýzy byla vytvořena síťová struktura složená z 5 virtuálních strojů Microsoft Windows Server 2012 R2 Standart. Tato architektura byla vytvořena na virtuálních zdrojích poskytovaných organizací Meta Centrum přes portál Openstack. Všech 5 strojů bylo přidaných do jedné sítě, která by se dala přirovnat například k LAN[1]. Vizualizace sítě je viditelná na obrázku 4.1.

Každý virtuální stroj má přidělenou IP adresu pro vnitřní adresování, přičemž portál poskytuje možnost alokace jedné IP adresy navíc, kterou je možné přidělit vždy jednomu ze strojů, a která slouží k adresaci v rámci Internetu, tedy pro vzdálené připojení.

K jednotlivým strojům je možné přistupovat přímo přes portál Openstack, který nabízí SPICE konzoli pro přímé ovládání stroje (obrázek 4.2), anebo přes zmíněnou volnou IP adresu, přičemž zde byl využit program Remmina pro připojení přes protokol RDP (Remote Desktop Protocol, obrázek 4.3).

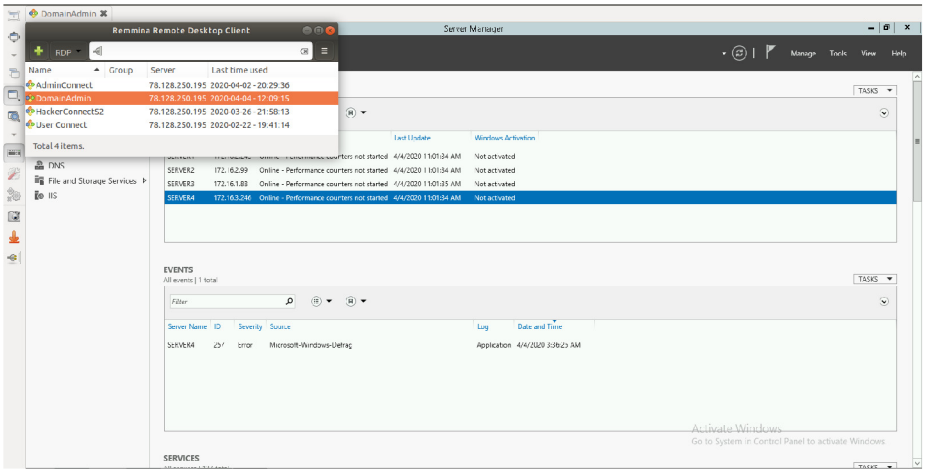


Obrázek 4.2: Snímek obrazovky konzole SPICE

Jak již bylo řečeno, síťová architektura se skládala ze 5 strojů. Čtyři z těchto strojů byly umístěny do stejné domény, která byla nazvána `project.bp` a byly spravovány pomocí adresářové služby Active Directory. Páty stroj byl umístěn mimo tuto doménu. Každý stroj měl určenou roli, v procesu analýzy síťového provozu.

- **SERVER 1** byl určený jako Domain Controler a zároveň DNS dotazovací server. Všechny ostatní servery v doméně byly nastaveny, aby se při autentizačních dotazech a také při překladu doménových jmen dotazovaly na tento server. Administrátor tohoto stroje byl zároveň super-administrátorem pro celou doménu. Zároveň byl z tohoto stroje vytvořen doménový účet běžného uživatele s názvem *User*, kterému ovšem nebyl povolen přístup k tomuto stroji. Stroj dostal přidělenou IP adresu 172.16.2.243.

[1] Local Area Network



Obrázek 4.3: Snímek obrazovky programu Remmina

- **SERVER 2** byl určený jako normální server, bez přidaných instalací. Pro tento server bylo přiděleno povolení k přístupu uživatelovi User. Tento server byl využíván jako cílový při útocích Pass the Ticket. Stroj dostal přidělenou IP adresu 172.16.2.99.
- **SERVER 3** byl určený jako server s nainstalovanými webovými službami. Byla zde vytvořena webová stránka dostupná pro doménu, která byla nastavena, aby vyžadovala autentizaci pro její zobrazení. Tato autentizace byla nastavena, aby probíhala pomocí protokolu NTLM. Na tento server byl také povolen přístup uživateli User. Tento server byl využíván jako cílový při útocích Pass the Hash.Stroj dostal přidělenou IP adresu 172.16.1.88.
- **SERVER 4** byl určený jako server, ze kterého byly prováděny útoky. Byl zde vytvořen lokální běžný účet Hacker, aby mohly být simulovány útoky jak z administrátorského, tak z běžného účtu. Stroj dostal přidělenou IP adresu 172.16.3.246.
- **SERVER X** byl určen jako server pro kontrolu správnosti navrhovaných řešení i pro stroje, které nenáleží do stejné domény. Stroj dostal přidělenou IP adresu 172.16.3.213.

Každý stroj byl obrazem kompletní instalace operačního systému Windows Server, takže obsahoval mnohem víc instalovaných nástrojů a jiných programů, které jsou implicitně dodávány společností Microsoft, a pro normální chod stroje v rámci síťové architektury používané například firmou by byly nezbytné, ovšem pro potřeby této práce byly zanedbány, protože nebyly pokládány za podstatné.

Pro potřeby práce byly implicitní bezpečnostní prvky systému (jako ochrana pomocí Windows Firewall nebo Internet Explorer Enhanced Security) omezeny na nejnižší možnou úroveň. To znamená, že Windows Firewall byla nastavena tak, aby byl propouštěn všechen provoz související s protokoly, které byly potřebné pro provádění potřebných úkonů a analýzy, spojené s vypracováním práce. U prohlížeče Internet Explorer byla Enhanced Security úplně vypnuta. Možný dopad těchto opatření je diskutován v závěru této práce.

Pro potřeby práce byly na servery instalovány následující aplikace třetích stran:

- **Wireshark** pro odchytávání síťového provozu.

- **Mozzila Firefox** pro pohodlnější a rychlejší přístup k internetovým a prohlížečovým službám, než které nabízí Internet Explorer

Nástroje potřebné pro provádění útoků nevyžadovaly instalaci, byly spustitelné jednoduše přes příkazový řádek.

4.2 Prerekvizity pro provádění útoků

Jak již bylo zmíněno, v této práci byla snaha detekovat útoky související se síťovými bezpečnostními protokoly z pohledu analýzy síťového provozu, tedy komunikace mezi počítači. Úkony odehrávající se na lokální úrovni, byly pro účely této práce zanedbatelné. Jinak řečeno, v procesu útoku na protokol byla v práci zkoumána pouze část začínající momentem, kde již útočník má potřebné údaje k útoku a přistupuje k útoku jako takovému.

Způsob, jakým útočník získá přístup k údajům není v této práci zkoumán, protože není možné ho nijak ovlivnit. V kapitole 3 byly popsány způsoby extrakce údajů použitelných pro útok, ovšem ty jsou pomocí sledování síťového provozu nedetekovatelné (buď jsou prováděny na lokální úrovni anebo jsou založeny na pouhém odposlechu síťové komunikace). Proto i při přípravě pro vlastní analýzu byly tyto údaje získávány tou nejjednodušší cestou. Způsob získání údajů je popsán dále.

Ukradení tiketu

Pro manipulaci s tickety byl v této práci využíván program *Rubeus* [2], který je volně dostupný na internetu přes portál GitHub. Rubeus je nástroj, vytvořený v jazyce C#, který je přímo určen pro manipulaci se surovými daty obsaženými v tiketech protokolu Kerberos.

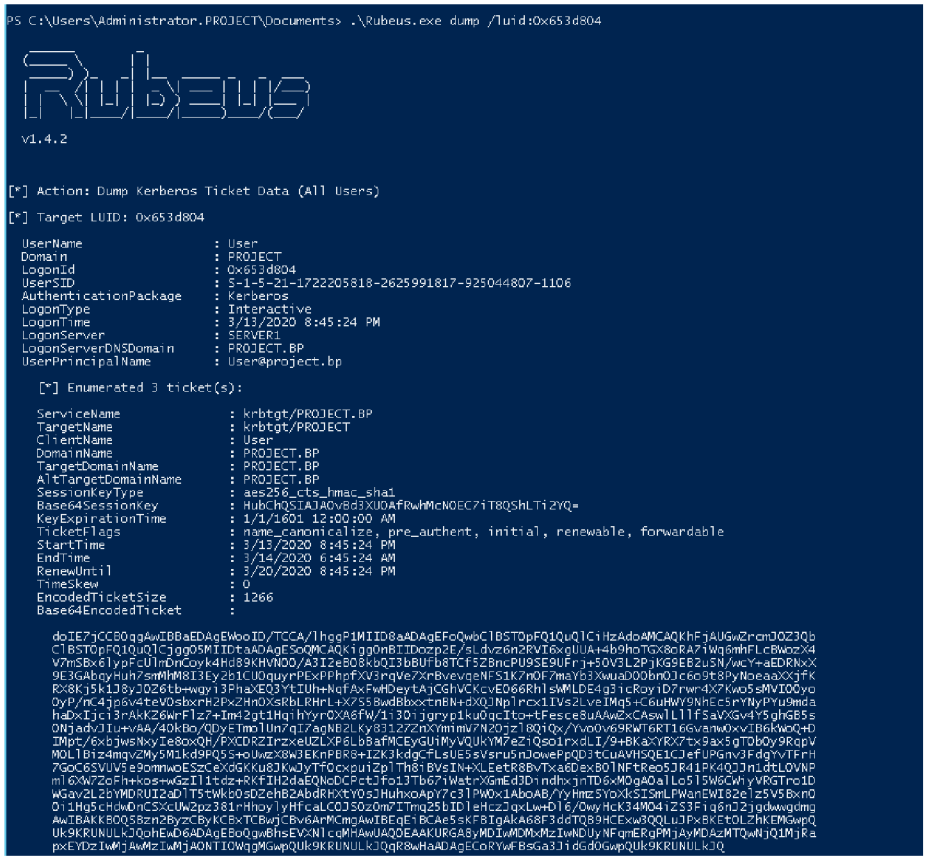
Pomocí tohoto nástroje je velmi lehké získat informace o všech tiketech, které má v dané době systém uložené v paměti, a je možné ho využít pro provádění útoků, které jsou popsány v sekci 3.1. Je zde nutné zmínit, že Rubeus pro svoje správné fungování požadoval doinstalování Microsoft .NET framework verze 3.5 na všechny stroje, kde byl spouštěn. Dále je také funkčnost nástroje omezena právy uživatele, který ho spouští, a proto bylo někdy nutné nástroj spouštět v příkazové řádce se zvýšenými oprávněními.

Postup, jakým byl extrahován tiket ze systému v rámci této práce, je následující. V systému, ke kterému měl přístup uživatel na jehož údaje útok cílil, bylo nejdříve pomocí příkazu `/triage` nástroje Rubeus zjištěno, jaké tikety jsou v systému uloženy. Následně pomocí příkazu `/dump /luid:[hex]` byly extrahovány požadované tikety ze systému. Parametr `luid` udává identifikaci tiketu, který má být extrahován. Výsledkem tohoto příkazu je tiket, kódovaný v Base64 [17]. Tento kódovaný tiket byl zkopírován a uložen pro další použití. Postup je viditelný na snímku obrazovky na obrázku 4.4. Extrahován byl tiket pro službu krbtgt, tedy Ticket Granting Ticket.

Následně na stroji, ze kterého probíhal útok byl pomocí příkazu `/ptt /ticket:[base64encoded-ticket]` nástroje Rubeus uložen předtím ukradený ticket k užívání. Postup je viditelný na obrázku 4.5.

Takto nastavený systém je připraven, aby přes něj byl proveden útok. Použitý postup byl převzat z webových stránek `tarlogic.com` [33] a `blog.stealthbits.com` [40].

[2] odkaz možno nalézt v [12]



Obrázek 4.4: Extrakce tiketu

Obrázek 4.5: Předání tiketu

Ukradení hash hodnoty hesla

Pro extrakci hash hodnoty hesel se SAM souborů byl použit nástroj *CQHashDumpv2* [19], vyvinutý organizací CQURE Academy pouze pro akademické účely zkoumání síťových útoků. Následná manipulace s hash hodnotou hesla byla prováděna pomocí nástroje *Mimikatz* [11], který je volně dostupný přes portál GitHub a nabízí velkou škálu možností pro manipulaci s autentizačními údaji v systému Windows.

Jak již bylo zmíněno, pomocí nástroje CQHashDumpv2 je možné přistoupit k obsahu SAM souborů, ve kterém jsou obsažena hesla (popsáno v sekci 3.2). Toto je však možné pouze při spuštění ze systémového účtu (uživatel `nt authority\system`). Pro přístup k tomuto účtu bylo využito nástroje `PsExec`, který je volně dostupný na internetu jako součást souboru nástrojů PsTools [21]. K systémovému účtu bylo přistoupeno spuštěním příkazového řádku pomocí tohoto nástroje zadáním příkazu `.\psexec.exe -s -i -d cmd.exe`, díky čemuž se spustila příkazová řádka pod systémovým účtem, jak je vidět na obrázku 4.6. Následně již mohl být použit nástroj CQHashDumpv2, kde pomocí příkazu `/samdump` byl vypsán lokální obsah SAM souboru s hashy uživatelských hesel, jak je možné vidět na obrázku 4.7. Potřebný hash hesla byl zkopírován a uložen pro další použití.

```
PS C:\Users\Administrator\Downloads\PSTools> .\PsExec.exe -s -i -d cmd.exe

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


cmd.exe started on SERVER3 with process ID 4272.
PS C:\Users\Administrator\Downloads\PSTools>
```

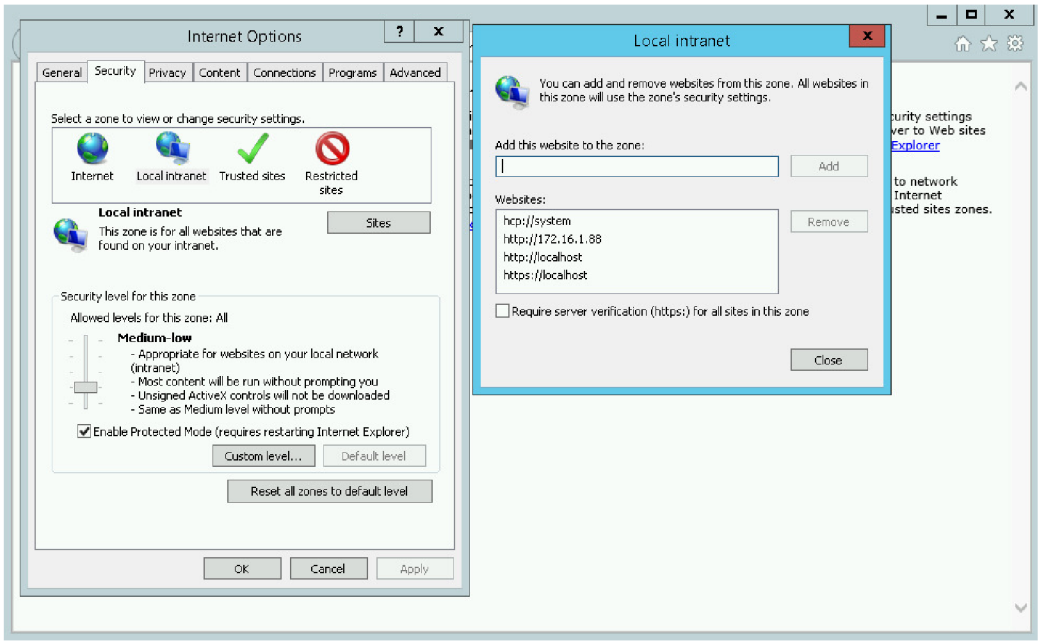
Obrázek 4.6: Spouštění systémového účtu

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\windows\system32>cd \Users\Administrator.PROJECT\Downloads\cqhashdumpv2

C:\Users\Administrator.PROJECT\Downloads\cqhashdumpv2>CQHashDumpv2.exe /samdump
SAM hashes:
Administrator:500:aad3b435b51404eeaad3b435b51404ee:731a4d5e5a2230fa43d74b4337131
4c7:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:731a4d5e5a2230fa43d74b43371314c7:::
cloudbase-init:1001:aad3b435b51404eeaad3b435b51404ee:50ead809cc7897cd922a4cf262b
d0d2d:::
Admin:1002:aad3b435b51404eeaad3b435b51404ee:a7482de99b16da318695a2164154c5a2:::
Hacker:1006:aad3b435b51404eeaad3b435b51404ee:19d41c868bf66fe5068c57e8c82ae512:::


C:\Users\Administrator.PROJECT\Downloads\cqhashdumpv2>
```

Obrázek 4.7: Extrakce hash hodnoty hesla

Na stroji, ze kterého měl probíhat útok, byl spuštěn nástroj Mimikatz. Nejdříve bylo nutné nastavit správná oprávnění nástroje. Takto bylo učiněno pomocí příkazu `privilege::debug`. Následně byla pomocí příkazu `sekurlsa::pth /User:[username] /ntlm:[hash]`

`/domain:[domain] /run:cmd.exe` spuštěna příkazová řádka, ve které byly uloženy přihlašovací údaje uživatele zadané v příkazu. Jak bylo popsáno výše, Server 3 byl určený jako cílový při útocích Pass the Hash. Zde běžely webové služby s nastavenou potřebou autentizace při žádostech o zobrazení webové stránky. Pro zobrazení webových stránek je nejvhodnější webový prohlížeč, proto byl z příkazové řádky, otevřené předchozím příkazem, spuštěn internetový prohlížeč Internet Explorer. Zde, aby mohla fungovat autentizace pomocí údajů nastavených předchozím příkazem, musela být žádaná stránka přidána do seznamu sítí lokálního intranetu, jak je možno vidět na obrázku 4.8. Takto nastavený systém byl připraven, aby z něho mohl být proveden útok. Použitý postup byl převzat ze stránek `cqureacademy.com` [19] a `labs.f-secure.com` [32].



Obrázek 4.8: Přidání cílové adresy do sítě intranet

4.3 Vlastní útok na architekturu

V předchozí sekci bylo popsáno, jakým způsobem byly získány údaje potřebné pro provedení útoků, které cílí na bezpečnostní protokoly. V této sekci je dále popsáno, jak byly tyto útoky prováděny, jak vypadal síťový provoz v době, kdy byly prováděny a dále také jaké informace byly získány ze sledování síťového toku.

V době útoku byl síťový provoz odchytáván pomocí nástroje Wireshark, přičemž ten byl spuštěn vždy na třech systémech – Domain Controller, systém na který by útok cílen a systém, ze kterého útok vycházel. Takto bylo zajištěno, že pro analýzu bude k dispozici kompletní síťový provoz, který je relevantní k dané situaci. V následujícím textu jsou přiloženy výřezy ze souborů `pcap`, které jsou důležité pro daný problém. Kompletní soubory jsou poté přiloženy na paměťovém médiu, popis obsahu části média se soubory se nachází v přílohách.

Na tomto místě je nutno zmínit, že přístup k analýze a detekci útoků, který je použit v této práci, není příliš rozšířený. Firma Flowmon, která práci zadávala, nebyla schopna poskytnout žádné materiály, podle kterých by bylo možné vypracovat přesný plán provádění

a sledování útoků na protokoly tak, aby byla reálná šance nějakého výsledku, který by mohl být dále použitelný při vytváření návrhu pro detekci útoků. Dohledané materiály, které souvisely s tématem, mohly být využity aspoň pro vytvoření hypotéz, jakým způsobem by mohlo být možné útoky detekovat, ale přesný plán, podle kterého by se analýza řídila, vypracován nebyl.

Všechna doposud popsaná nastavení sítě, včetně prerekvizit pro útok, byla zvolena tak, aby z nich mohlo vycházet co největší množství případů užití. Byla podporována zavedenými zvyklostmi (jako například, že v každé doméně bude jeden Domain Controller, který nejspíše zajišťuje autentizační autoritu, nebo použití nástrojů pro útok, které jsou více známé a u kterých je větší možnost, že budou použity při útoku v reálné situaci), tedy existoval alespoň lehký návod, jak postupovat, aby možný výsledek pokrýval co největší oblast situací, ke které může být relevantní.

Útoky jsou dále popisovány vždy podle jejich typu tak, jak byly prováděny a analyzovány, dokud nebylo možné vyvodit ze získaných informací nějaký závěr.

Pass the Ticket

Po přípravné fázi byl stroj, ze kterého byl prováděn útok, ve stavu, kdy měl v paměti uložený ticket, ukradený z jiného stroje, a byl připraven ho uplatnit při autentizaci pomocí nástroje Rubeus.

Při hledání služby, která by používala pro svou autentizaci protokol Kerberos, naštěstí nebylo nutné vymýšlet nic zvláštního, protože tento protokol je používán implicitně při autentizaci uživatelů v rámci sítě. Stačilo tedy ze stroje, ze kterého byl prováděn útok, požádat o přístup ke stroji, který se měl stát terčem útoku. Jak již bylo zmíněno, pro útok byl používán nástroj Rubeus, což je konzolová aplikace. Útok tedy nemohl být veden například přes protokol vzdálené plochy (RDP), který nebývá spouštěn z příkazové řádky. V úvahu připadal protokol SSH, ovšem ten není implicitně podporován na systémech Windows, musí být vždy doinstalován. To znamená, že na strojích, které jej výslovně nepotřebují, nejspíše k dispozici nebude, a tudíž přes něj nebude tak často útok veden. Nakonec byl zvolen útok pomocí příkazu `Enter-PSSession`, což je cmdlet dostupný ve Windows Power Shell, který vytvoří novou instanci Power Shell na definovaném stroji.

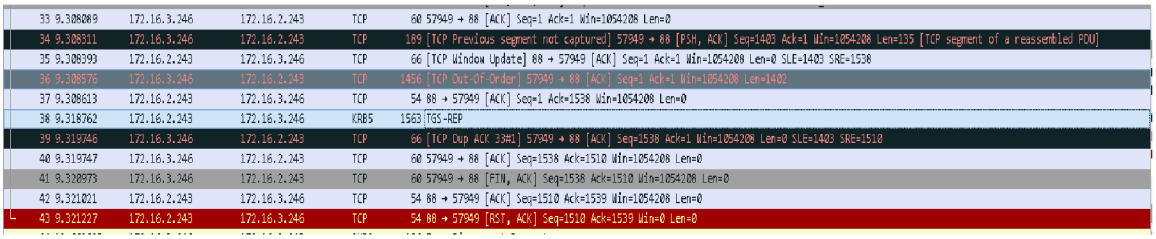
```
PS C:\Users\Hacker\Downloads> Enter-PSSession -ComputerName server2.project.bp -Authentication kerberos
[server2.project.bp]: PS C:\Users\Administrator.PROJECT\Documents> whoami
project\administrator
```

Obrázek 4.9: Útok Pass the Ticket

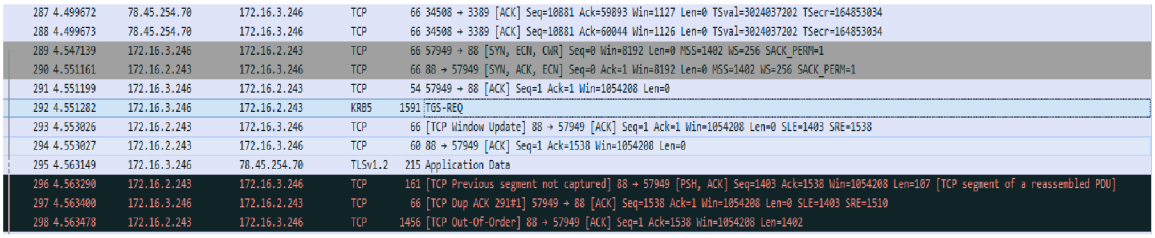
Kompletní příkaz použitý pro útok, který byl vedený ze standardního účtu, byl `Enter-PSSession -ComputerName server2.project.bp -Authentication kerberos`. Potupně byl proveden útok s pomocí ukradeného ticketu, který náležel uživatelům User a doménový administrátor. Bylo to z důvodu, kdyby byl rozdíl v autentizace standardního a a uživatelského účtu. V obou případech šlo o úspěšný útok, jehož vizualizaci pro administrátorský účet je možno vidět na obrázku 4.9.

Co se týče síťového provozu, nebyl žádný rozdíl mezi autentizací u standardního a administrátorského účtu, proto mezi nimi není dále rozlišováno. Na obrázcích 4.10, 4.11 a 4.12 je vidět výřez z souborů PCAP, ve kterých byl zachycen síťový provoz v době útoku. Výřez obsahuje pakety, které mají přímou souvislost se síťovou autentizací.

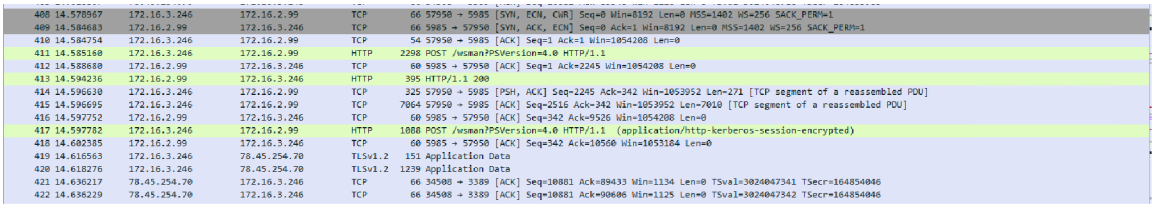
Na obrázcích je možno vidět postup autentizace přesně tak, jak byla popsána v sekci 2.3. Stroj, ze kterého je útok prováděn, žádá o client-to-server tiket u TGS (ticket-granting se-



Obrázek 4.10: Komunikace mezi útočícím strojem a DC při útoku Pass the Ticket



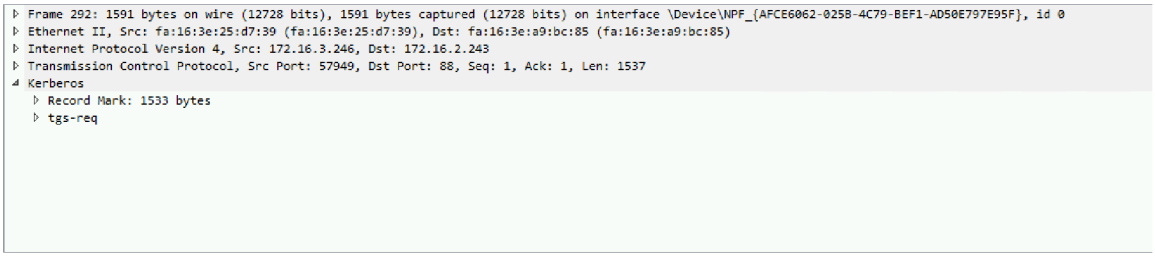
Obrázek 4.11: Komunikace mezi útočícím strojem a DC při útoku Pass the Ticket



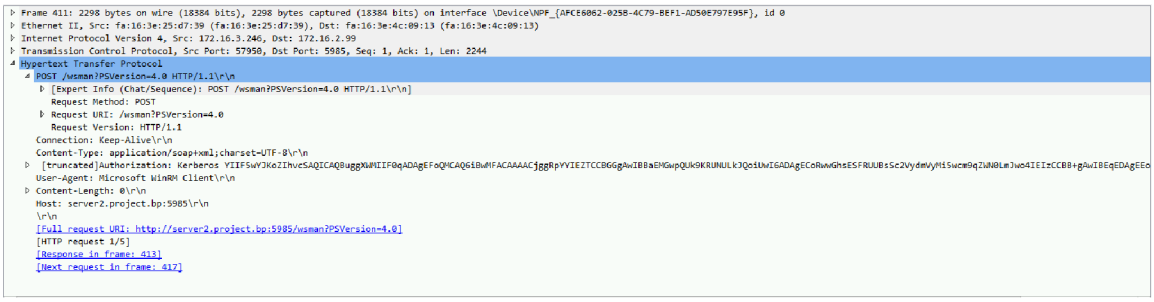
Obrázek 4.12: Komunikace mezi útočícím a cílovým strojem při útoku Pass the Ticket

ver, který je zároveň i Domain Controller), kde se autentizuje pomocí TGT (ticket-granting ticket). Toto je zajištěno zprávou TGS_REQ. Server odpovídá zprávou TGS_REP, ve které je obsažen žádaný tiket, zajišťující umožnění přístupu ke službě na daném serveru. Následně, za využití HTTP protokolu, začíná komunikace mezi útočníkem a cílovým systémem. V první zprávě přiloží útočníkův stroj obdržený client-to-server ticket, díky němuž se autentizuje vůči danému serveru, následná komunikace již probíhá šifrovaně.

Obrázky 4.13 a 4.14 obsahují detail paketů, které zajišťovaly autentizaci vůči serverům, tedy paketů, které byly zasílány ve směru od útočníka jinam. Prvotní předpoklad byl, že pokud bude v síťovém provozu nějaká anomálie, bude souviset s pakety, které odesílá útočící stroj, a proto ty byly zkoumány primárně.



Obrázek 4.13: Paket TGS_REQ odchycený při útoku Pass the Ticket



Obrázek 4.14: Autentizační HTTP paket odchycený při útoku Pass the Ticket

Aby mohla být provedena úplná analýza, byl odchytáván také síťový provoz v čase, kdy byl stejný postup, jaký je popsán výše, proveden legitimní cestou, tedy autentizace proběhla ne pomocí ukradeného tiketu, ale pomocí správné kombinace uživatelského jména a hesla. Zkoumané pakety z provozu v době útoku, ale ze zjevných důvodů i pakety putující jako jejich odpovědi, byly porovnány s těmi, které byly zachyceny při normální autentizaci a pro úplnost také se vzorovými, které byly nalezeny na internetu (stránka `medium.com` [5]).

Pass the Hash

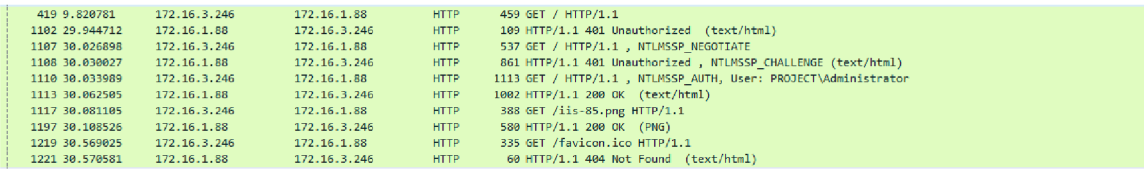
Po přípravné fázi byl na stroji spuštěn a nakonfigurován webový prohlížeč Internet Explorer tak, aby využíval pro autentizaci předem dané uživatelské jméno a hash hesla.

Jak již bylo zmíněno, protokol NTLM je stále podporován systémem Windows a v některých případech aktivně využíván. Pokud je dostupný protokol Kerberos, tak dostává automaticky přednost, ovšem starší systémy nebo aplikace třetích stran mohou mít implementovanou pouze autentizaci pomocí NTLM. Stejná situace může nastat také u webových služeb a toho bylo v této práci využito. Jak bylo zmíněno výše, na jednom ze systémů byly

nainstalovány webové služby a byla tam spuštěna webová stránka vyžadující autentizaci. Tato autentizace byla nastavena tak, aby probíhala pouze přes protokol NTLM.

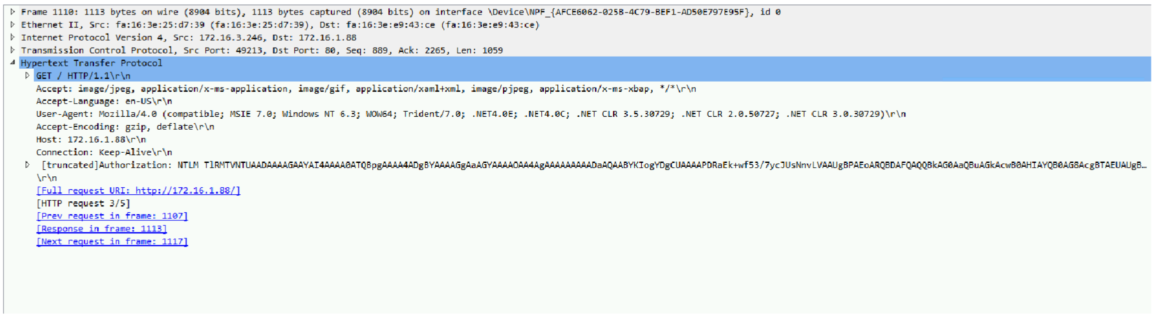
Útok byl proveden pouhým zadáním adresy, která byla asociována na serveru s běžícími webovými službami s webovou stránkou, do adresní řádky prohlížeče. Prohlížeč následně zobrazil dotazovanou webovou stránku, což znamená, že útok proběhl úspěšně.

Na obrázku 4.15 je výřez PCAP souboru, který obsahuje zachycený provoz v době, kdy probíhal útok. Je zde možno vidět komunikaci, která odpovídá autentizaci, jak byla popsána v sekci 2.3. Nejdříve odeslal dotazující stroj pomocí HTTP protokolu negotiate zprávu. Server odpověděl zprávou obsahující challenge. Na tu útočící stroj odpověděl zprávou obsahující response. Komunikace mezi webovým serverem a Domain Controllerem probíhala zašifrovaně a proto není zobrazena.



Obrázek 4.15: Odchycená komunikace při útoku Pass the Hash

Na obrázku 4.16 je zobrazen detail paketu se zprávou response. Předpoklad byl, stejně jako u útoku Pass the Ticket, že případná anomálie se bude nacházet v paketech jdoucích ve směru od útočícího stroje. Stejně, jako v předchozím případě, byl i zde odchytáván provoz v době, kdy autentizace probíhala legitimní cestou. S pakety zachycenými v tomto toku, byly porovnávány ty zachycené během útoku.



Obrázek 4.16: Paket s response zprávou protokolu NTLM

4.4 Zhodnocení výsledků sledování a návrh detekce útoků

V této sekci jsou blíže zkoumána data, získaná při sledování síťového provozu v době útoku. Jak již bylo zmiňováno několikrát, v této práci je pohlíženo na analýzu útoků pouze z pohledu síťového provozu, z důvodu cíle práce, kterým je vytvoření pluginu pro exportér síťového toku. Ovšem taková síťová struktura, která byla vytvořena pro účely zkoumání útoků, a tedy taková, jakou by bylo možné nalézt ve velké firmě, vytváří systém, ve kterém by rozhodně byla chyba ignorovat vše ostatní, kromě síťové komunikace.

Pokud by z procesu autentizace, jak ho provádí v této práci zkoumané protokoly, byly vybrány pouze kroky související se síťovou komunikací, zůstala by nejspíše přibližně polovina z kroků. To znamená, že velká část autentizačního procesu probíhá na lokální úrovni,

například jako výpočet nějaké hodnoty, porovnání, vytváření zprávy. Všechna tato aktivita může být nějak zaznamenávána, a tudíž nějakým způsobem kontrolována. Pokud tedy je základním objektem analýzy v této práci síťový provoz, může být využit, v nejoptimálnějším případě, jako přímý indikátor nelegitimní aktivity, a tedy útoku, anebo mohou být informace z něj získané využity pro zjištění aktivity, která se může jevit jako podezřelá, a předány ke kontrole jiným zdrojům, například lidským.

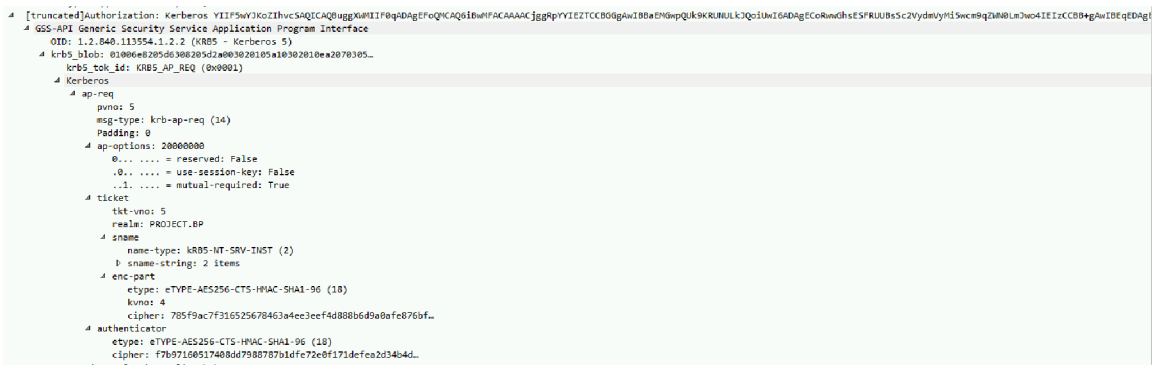
Návrhy detekce útoků v této sekci vychází z takovýchto předpokladů a při jejich vytvoření bylo využíváno i informací, dostupných z jiných strojů, než ze síťového toku, ale souvisejících s problémem.

Sledování a detekce Pass the Ticket

Před zahájením popisu analýzy tohoto útoku je nutné připomenout, na jakých principech staví protokol Kerberos. Hlavní myšlenkou tohoto protokolu je kontrola totožnosti u nějaké autority. Stroje využívající tento protokol mají uloženu informaci, kdo je pro ně touto autoritou a u ní kontrolují identitu a přístupová práva uživatele, který žádá přístup k jejich službám. Prokazování identity probíhá pomocí takzvaných tiketů, které v sobě mají mimo jiné informaci, pro jakého uživatele jsou vydávány. Uživatel, který chce tiket použít, k němu přikládá jím vytvořenou zprávu, ve které prokazuje svoji totožnost. Úspěšné použití tiketu předpokládá shodnou identitu uživatele, pro kterého byl tento tiket vydán, s uživatelem, který tiket použil.

Se znalostí výše popsaného principu je možné si vytvořit hypotézu nejjednoduššího způsobu detekce útoku Pass the Ticket. Tento útok zakládá na použití tiketu, který náleží jinému uživateli. Nejjednodušeji by šel útok zjistit tak, že se porovná uživatel využívající tiket s uživatelem uloženým přímo v tiketu. Samozřejmě, že pokud by authenticator, tedy zpráva dokládající identitu uživatele, který tiket využívá, byl vytvořen útočníkem a byla by v něm uvedena jeho identita, tak by útoku zabránil samotný protokol. Protože byl útok v pokusech prováděných v této práci úspěšný, aniž by byl jakkoliv manuálně upravován authenticator, dá se předpokládat, že toto zajišťuje automaticky sám nástroj použitý pro útok a tudíž není těžké authenticator zfalšovat a je to obvyklá praxe útočníků.

Na obrázku 4.17 je možno vidět detail části HTTP paketu, kterým se autentizoval útočník při útoku, obsahující informace pro protokol Kerberos.



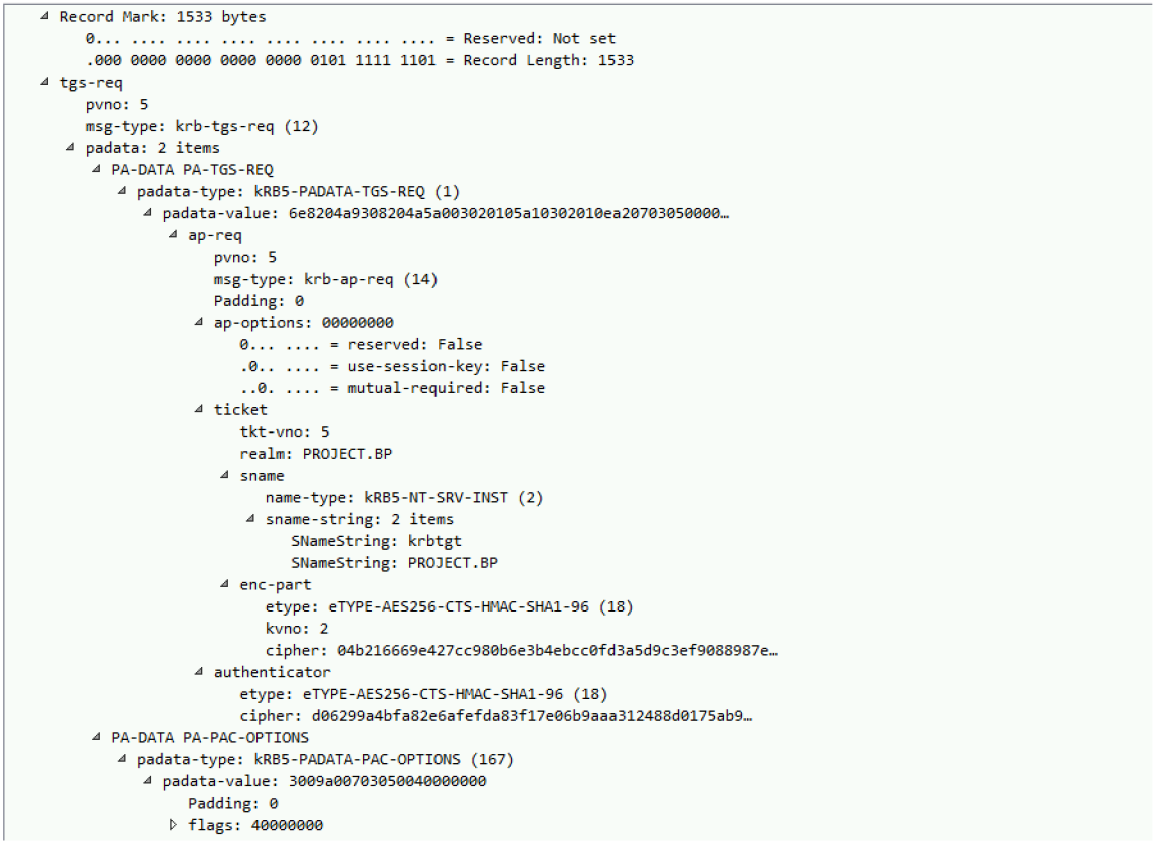
Obrázek 4.17: Detail autentizačního HTTP paketu zachyceného při útoku Pass the Ticket

Pro účely analýzy jsou důležité pouze části `ticket` a `authenticator`, předchozí informace pouze určují typ zprávy a další parametry. V části tiketu je možné vidět doménu, pro kterou byl tiket vydán, dále protokol a server, kde je možno tiket uplatnit. Zbytek

části, kde se znalostí protokolu můžeme předpokládat název služby, identitu uživatele a jiné informace, například dobu platnosti a možnosti obnovení tiketu, je zašifrovaný. Stejně tak je zašifrovaný authenticator.

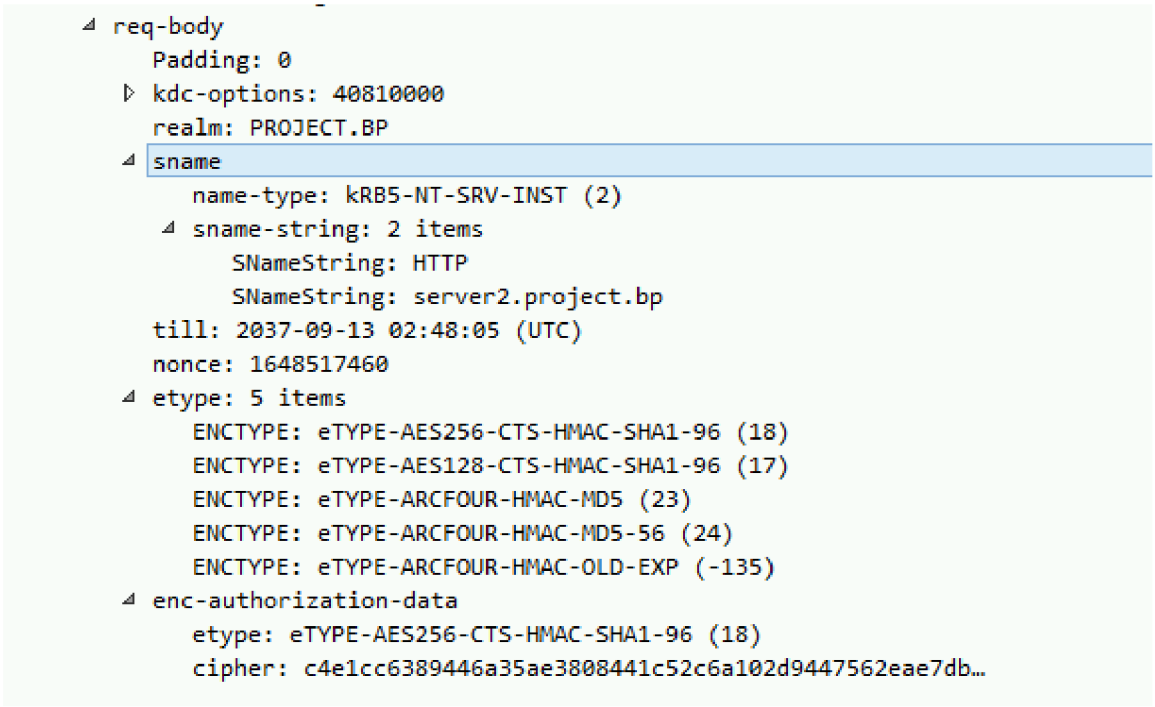
Z pohledu analýzy síťového provozu je možno tvrdit, že zašifrované data jsou nepoužitelná. Nejde předpokládat dostupnost klíče pro rozšifrování, protože pokud by klíč byl dostupný nástroji sledující provoz, nejspíše by šlo o příliš velké bezpečnostní riziko. Pokud je tedy ignorována zašifrovaná část, zůstávají informace, které nejsou pro účely návrhu detekce útoku nijak využitelné.

Pouze z informací z jednoho paketu, který byl použit k autentizaci při přístupu ke službě tedy útok detekovat nelze. Dále se naskýtá možnost, zjistit nelegitimní žádost, ve které jde vůbec o vydání tohoto tiketu. V sekci 4.3 je u popisu útoku Pass the Ticket zobrazena komunikace mezi útočícím strojem a ticket-granting serverem, kde je možno vidět zprávu TGS_REQ. Detail této zprávy po částech je na obrázcích 4.18 a 4.19.



Obrázek 4.18: Detail zprávy TGS_REQ zachycené při útoku Pass the Ticket

Ze znalosti protokolu Kerberos vyplývá, že součástí této zprávy bude ticket-granting ticket. Bohužel, stejně jako v předchozím případě, jsou jediné nezašifrované údaje ty o doméně a cílovém serveru. Ostatní data jsou informace protokolu Kerberos, například o použitém šifrování. Zde dokumentace použitého nástroje, což byl Rubeus, zmiňuje největší slabost tohoto nástroje, kterým je použitá šifrovací metoda. Moderní systémy Windows používají šifru AES, ovšem Rubeus by při útoku měl používat šifru RC4_HMAC. Při porovnání dat z toků získaných při útoku a při klasickém přihlášení se informace o použitém šifrování nelišila, tudíž není možné takto útok detekovat.



Obrázek 4.19: Detail zprávy TGS_REQ zachycené při útoku Pass the Ticket

Ani při bližším pohledu na TGS_REQ zprávu nebylo možné vytvořit žádnou hypotézu, jak by mohl být útok detekován. V tuto chvíli nastal čas na rozšíření oblasti, ze které byly informace získávány. K tomuto byla velmi nápomocná webová stránka `blog.stealthbits.com`. Článek *How To Detect Pass-the-Ticket Attack* [40], který se na ní nachází, obsahuje návrhy a myšlenky, které by mohly pomoci k jednoduché detekci tohoto útoku, pracuje však pouze se zkoumáním systémových logů. Každá autentizační činnost systému je zaznamenávána a uložena. V případě systémových logů je poněkud jednodušší útok detekovat, neboť oproti síťovému toku v sobě obsahují více informací, které jsou nezašifrované, ale vzhledem k tomu, že jde o informace zpracovávané a ukládané lokálně, jim v této práci nebude věnována větší pozornost. Myšlenky použité při detekci útoků pomocí kontroly logů lze ovšem použít i v této práci.

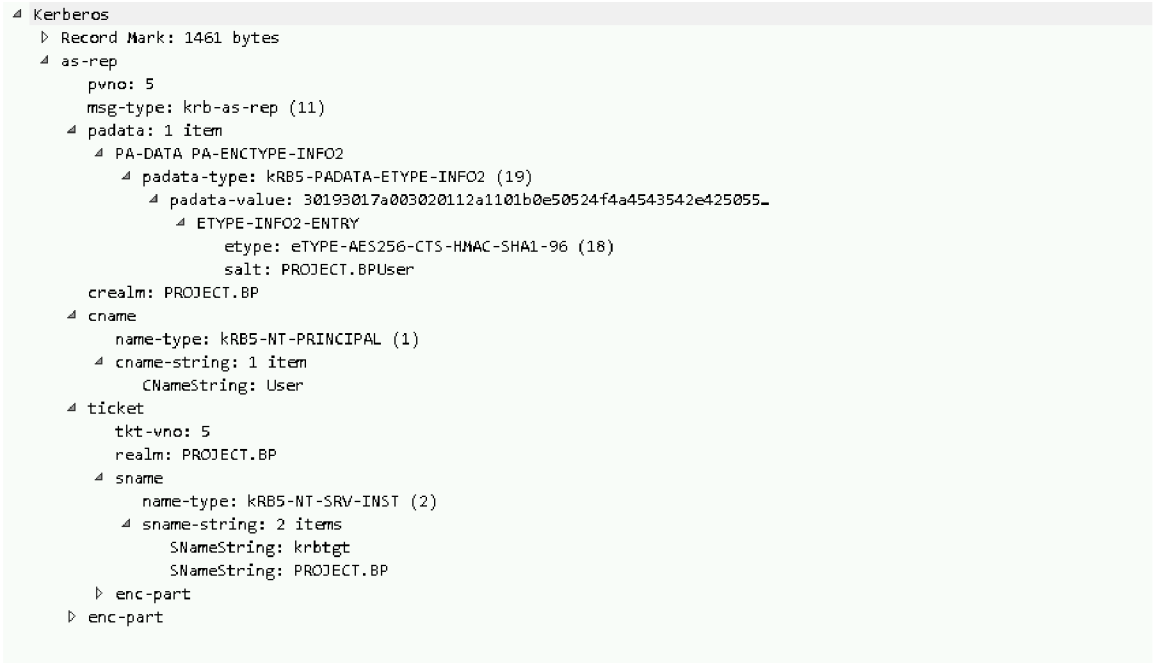
Pro přehlednost je zde znovu shrnuto, jak by měla vypadat komunikace z pohledu uživatelova stroje, když se snaží autentizovat a nejsou pro něj uloženy žádné tikety.

- Uživatel zadá svoje autentizační údaje.
- Stroj odešle zprávu AS_REQ, požadující vystavení nebo obnovení TGT, očekává AS_REP.
- Po obdržení stroj odešle zprávu TGS_REQ, požadující vystavení nebo obnovení client-server ticketu, očekává zprávu TGS_REP.
- Po obdržení využije stroj tiket k přístupu ke službě.

Při porovnání uvedeného postupu a reálného provozu, který byl odchycen v době útoku, zde vyplyne jedna rozdílnost. V komunikaci chybí kombinace zpráv AS_REQ a AS_REP. Toto znamená, že stroj již měl uložený TGT, a tudíž o něj nemusel žádat. To není nijak

neobvyklé, příklady takovéto komunikace je možné odchytit v jakémkoliv provozu protokolu Kerberos. Pokud je TGT uložen, stroj nežádá o jeho vystavení, pouze pokud vyprší jeho platnost, žádá o jeho obnovení. Z pohledu síťového provozu to znamená, že v době před použitím TGT, která bude maximálně rovna jeho platnosti, by se měla v síťovém provozu objevit žádost o jeho vystavení nebo obnovení. Pokud tomu tak není, jde o podezřelé chování, znamená to totiž, že byl tiket nejspíše do stroje uměle vložen, a tedy že se jedná nejspíše o útok.

Důležité je ještě ověřit, zda je toto zjistitelné ze sítě. Protokol Kerberos je nastaven tak, aby vždy vydal jeden tiket určitého druhu jednomu uživateli. Z předchozích detailů autentizačních zpráv je jasné, že jednotlivé zprávy je možno rozlišit. Ještě je potřeba ověřit, zda je v z těchto zpráv možno zjistit identitu uživatele, kterému je tiket vystavován.



Obrázek 4.20: Detail zprávy AS_REP zachycené při útoku Pass the Ticket

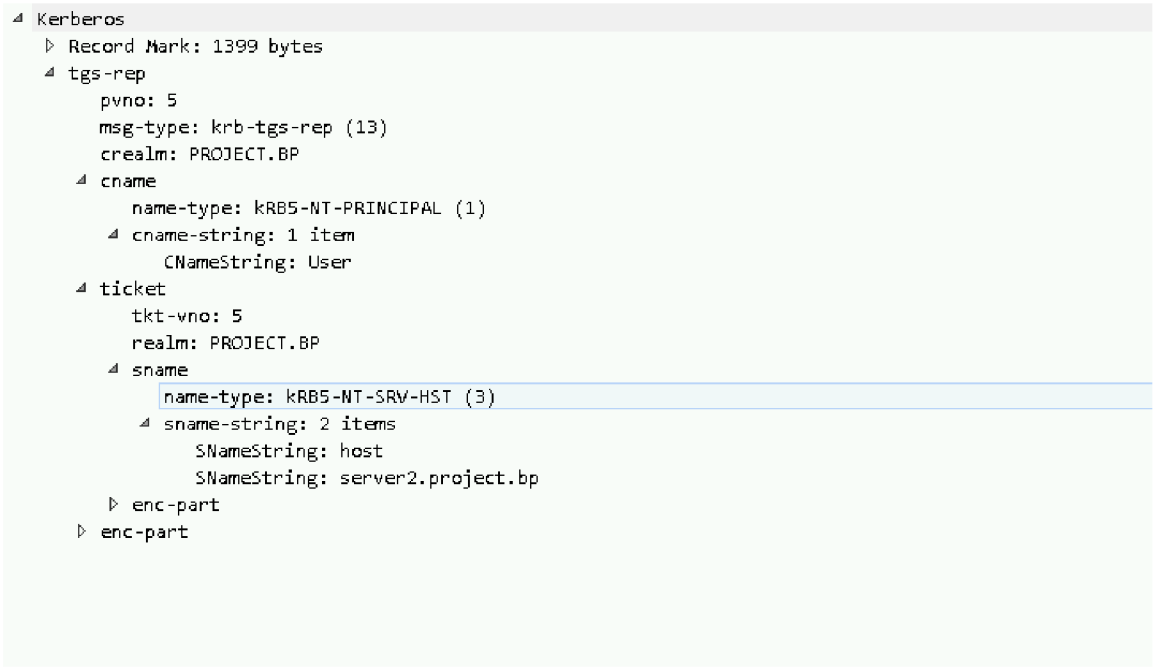
Na obrázcích 4.20 a 4.21 je možno vidět, že jméno uživatele je ve zprávě obsaženo. Popsaný způsob je tedy bez problémů použitelný.

Je nutno zmínit, že výše uvedeným způsobem není možno stoprocentně určit, jestli se jedná o útok. Může však být při takovéto situaci upozorněno na velice podezřelou síťovou aktivitu, která může být předána k další kontrole.

Konkrétní způsob detekce útoku

Způsob, jak detekovat útok Pass the Ticket navrhovaný touto prací je následující. Po dobu určenou nastavením Kerberosu ohledně platnosti jednotlivých TGT sledovat provoz v síti a ukládat informace o tiketech vystavených pro jednotlivé uživatele. Ve chvíli, kdy je jisté, že v síti nemůže být žádný platný tiket, o kterém by nebyl záznam, zahájit kontrolu způsobem, jaký byl popsán výše.

Plugin, vytvářený v této práci pro exportér, získává ze síťového toku tyto informace o tiketech a dále je předává ke zpracování.



Obrázek 4.21: Detail zprávy TGS_REP zachycené při útoku Pass the Ticket

Sledování a detekce Pass the Hash

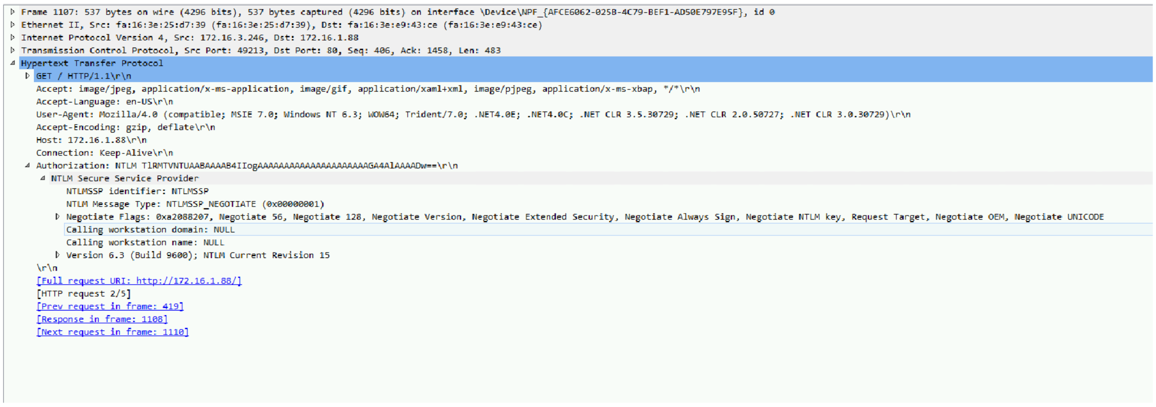
Stejně jako u předchozího útoku i zde je pro jistotu lepší připomenout způsob, jakým probíhá autentizace pomocí protokolu NTLM. Tento protokol zajišťuje autentizaci uživatele pouze pomocí porovnání hodnoty, která je vypočítávána z jeho autentizačních údajů, blíže tedy z jeho uživatelského hesla, s hodnotou uloženou v databázi.

Samotná podstata útoku Pass the Hash bohužel nenabízí moc možností, jak útok spolehlivě detekovat. Princip celého útoku spočívá pouze v přeskočení jednoho kroku procesu autentizace, a to přepočtu hesla zadaného uživatelem na jeho hash hodnotu, z důvodu, že je zadávána přímo hash hodnota. Útok tedy může být detekován, pokud existuje spolehlivý způsob, jak určit zadání hash hodnoty místo hesla.

Z pohledu sledování síťové komunikace je s jistotou možno prohlásit, že takovýto způsob neexistuje. K tomu není ani nutný pohled na odchycenou komunikaci protokolu, vystačí pouze znalost přesného postupu, jaký je v protokolu uplatňován. Heslo je na hodnotu hash přepočítáváno na lokální úrovni ještě před tím, než je zahájena síťová komunikace. Proto z pohledu sledování síťového provozu není žádný rozdíl mezi tím, když je zadáno heslo nebo rovnou jeho hodnota hash.

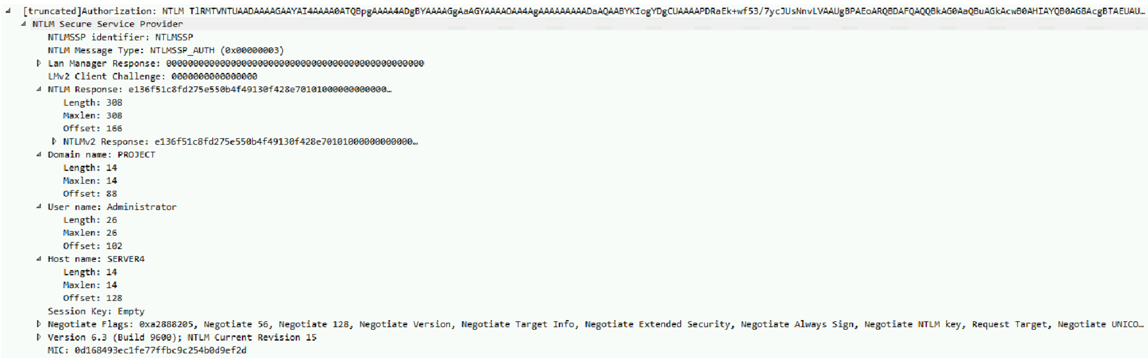
Na následujících obrázcích 4.22 a 4.23 jsou detaily autentizační části paketů obsahujících žádost klienta o autentizaci a poté jeho odpověď na výzvu serveru obsahující *challenge*.

Další možnost jak detekovat útok by bylo zjistit, zda uživatel, jehož údaje jsou využívány, je ten stejný, který je zadává. Na obrázku 4.22 nejsou žádné informace o uživateli, pouze vnitřní zprávy protokolu. Na obrázku 4.23 jsou již obsaženy informace o uživateli, jehož přihlašovací údaje jsou využívány, blíže tedy jeho jméno, doména a stroj, ze kterého se přihlašuje. Dále také můžeme vidět zprávu response. Toto jsou však všechny informace, které by bylo možné z paketu nějak využít a ty rozhodně nejsou využitelné k ověření identity uživatele, který přihlašovací údaje využívá.



Obrázek 4.22: Detail zprávy negotiate protokolu NTLM

Stejně jako u předchozího útoku i zde byla rozšířena oblast, ze které byly informace získávány o systémové logy, jak navrhuje článek *How To Detect Pass-the-Hash Attacks* na webové stránce `blog.stealthbits.com` [41]. Tento článek obsahuje zajímavé myšlenky ohledně čtení systémových logů v souvislosti s útokem Pass the Hash. Jak již však mírně vyplývá z informací diskutovaných výše, protože hlavní část útoku se odehrává na lokální úrovni stroje, ze kterého je útok veden, jednoznačně lze útok určit pouze z logů tohoto stroje. Obecně je velmi těžké, kontrolovat neustále logy všech strojů v síti, zda se v nich neobjeví kombinace logů, která by indikovala útok, ovšem skutečnost, že detekce je na lokální úrovni možná, může být v této práci využitelná.



Obrázek 4.23: Detail zprávy response protokolu NTLM

V rámci autentizační zprávy zasílá protokol také informace o stroji, ze kterého zpráva vychází, což je jeho jméno. Pokud je uvažována síť v rámci nějaké organizace nebo firmy, může platit předpoklad, že se jména strojů budou co nejméně měnit. Další platný předpoklad je, že uživatelé nebudou příliš často střídat stroje, které používají. Obvykle budou využívat buďto stroj, který se nachází na jim přiděleném místě, anebo vlastní přenosný. V obou případech budou jejich žádosti o autentizaci vycházet stále ze stoje se stejným jménem.

Konkrétní způsob detekce útoku

Na výše popsaném principu lze postavit ne přímo způsob detekce útoku Pass the Hash, ale rozhodně způsob detekce podezřelé aktivity. V autentizační zprávě je explicitně uvedeno

jméno uživatele a stroj, ze kterého požadavek vychází. Pokud kombinace těchto dvou údajů bude nezvyklá, tedy neodpovídající předchozímu dlouhodobému trendu, který je nutné nějakou dobu sledovat, může být vytvořeno upozornění o nezvyklé aktivitě následované, dle možností, kontrolou logů na zdrojovém stroji anebo upozorněním uživatele přes komunikační kanál, že byly jeho přihlašovací údaje použity z dříve neznámého zdroje.

Plugin, vytvářený v této práci pro exportér, získává ze síťového toku informace popsané výše a dále je předává ke zpracování.

Kapitola 5

Vytvoření pluginů pro exportér Flowmon sondy

Náplní finální části této práce bylo vytvoření pluginu pro exportér Flowmon sondy. Jak již bylo zmíněno v kapitole 2, v exportéru jsou přítomny vždy 4 typy pluginů. Cílem této práce bylo vytvořit pro každý zkoumaný protokol takový procesní plugin, aby bylo možné s jeho pomocí ze sledovaného provozu na síti vyextrahovat informace potřebné k následné detekci útoků pomocí způsobů, které byly popsány výše.

Pro každé vytvořené řešení, popsané v kapitole 4, byl tedy vytvořen odpovídající plugin, který prochází pakety zachycené exportérem ze síťového provozu a kontroluje přítomnost zpráv souvisejících s protokolem, který zpracovává. V případě zjištěné přítomnosti zprávy daného protokolu v paketu plugin ve zprávě hledá žádané informace, které následně předává dál.

5.1 Návrh architektury a implementace pluginů

Základní architektura každého pluginu je dána rozhraním exportéru a funkcemi, které musí být pro správné fungování pluginu přítomny. Každý plugin musí při své inicializaci zaregistrovat v exportéru takzvané '`hooks`', což jsou v principu informace pro exportér, kterým pluginům má předávat pakety ze síťovém toku po jejich zachycení. Existuje několik typů `hooks`, například spouštějící se při zachycení nového toku, při každém dalším zachycené paketu již zaznamenaného toku anebo po skončení toku. Plugin, po tom co obdrží zachycený paket v závislosti na tom, jaký `hook` má zaregistrovaný, z tohoto paketu získá potřebné informace a poté opět volá funkce exportéru, aby tyto informace předal dál.

Firma Flowmon poskytuje pro vývoj pluginů dokumentaci, ve které se nachází obecný popis fungování exportéru, základní postup jak vytvářet nový plugin a příklad ve formě procesního pluginu zpracovávajícího protokol SSH, kde poskytuje kompletní zdrojový kód tohoto pluginu a všech knihoven funkcí exportéru, které využívá tento nebo jiné exportní pluginy.

Architektura pluginů vyvíjených v rámci této práce byla tedy založena na příkladovém pluginu, který zpracovává protokol SSH. Dle tohoto příkladu byla vytvořena struktura pluginu typu `flowmon_plugin_t` (výpis z kódu 1), která byla registrována v exportéru. Základní funkce v pluginu jsou pro inicializaci a pro vypnutí. V rámci inicializace plugin vždy alokuje potřebný prostor pro své fungování a pomocí funkce `flowmon_attribute_add` (výpis z kódu 2) volané pro každý atribut, který bude ze zpracovávaného toku plugin

extrahovat, registruje tyto atributy. Plugin v tuto dobu také provádí registraci `hooks` (výpis z kódu 3).

```
1     static flowmon_plugin_t plugin = {
2         .type = FLOWMON_PLUGIN_TYPE_PROCESS,
3         .name = "kerberos",
4         .help_message = "kerberos protocol parser\n",
5         .flow_addon_size = sizeof(plugin_record_t),
6         .flags = FLOWMON_PLUGIN_FLAG_NEED_PACKET,
7         .init = process_kerberos_init,
8         .init_queue = process_kerberos_init_queue,
9         .shutdown_queue = process_kerberos_shutdown_queue,
10        .shutdown = process_kerberos_shutdown,
11        .rpc = NULL,                                          /
12      };
```

Listing 1: Struktura pluginu

```
1     retval->c_name_string  = flowmon_attribute_add(getter_list,
      ↪  "KERBEROS_CLIENT_NAME", 0, RECORD_VALUE_TYPE_STRING,
      ↪  RECORD_GETTER_GROUP_ID_NEW);
```

Listing 2: Přidání atributu pro export

```
1     flowmon_hook_queue_add(retval, plugin_id, process_kerberos_update);
2     flowmon_hook_queue_update(retval, plugin_id, process_kerberos_update);
```

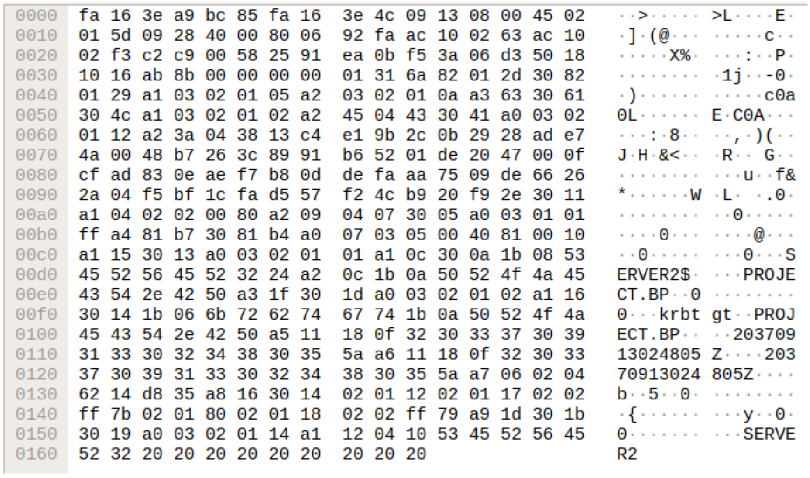
Listing 3: Registrace hooks

Toto chování je identické pro všechny vyvíjené pluginy. V následujícím textu jsou popsány jednotlivé navržené a realizované implementace, jaké byly u pluginů dále použity pro zpracování daných protokolů.

Plugin pro Kerberos

Dle RFC 4210 je pro vytváření síťové zprávy protokolu Kerberos využíváno kódování ASN 1. DER(Abstract Syntax Notation One Distinguished Encoding Rules) [15]. Způsob tohoto kódování je popsán v příloze C, která byla převzata z práce Ing. Petra Matouška, PhD., Description of IEC 61850Communication. V této příloze je popsáno kódování BER (Basic Encoding Rules), které se ovšem dle X.690 [39] od kódování DER přílišně neliší, DER kódování pouze upřesňuje principy BER kódování. Zkráceně popsáno, princip tohoto kódování využívá kombinaci trojice hodnot, identifikátoru, délky a hodnoty, kdy hodnota může být složena z dalších trojic.

Při návrhu implementace rozebrání zprávy byla pro inspiraci využita dokumentace [43] a zdrojové kódy [38] programu Wireshark, který už má dané filtry implementované.



Obrázek 5.1: Syrová data zprávy protokolu Kerberos

Na obrázku 5.1 je možno vidět syrová data zprávy protokolu Kerberos typu AS_REQ, jak byly odchyceny ze sítě. Zobrazena jsou způsobem, jakým to umožňuje program Wireshark, tedy hexadecimální hodnoty obsažené v bajtech ve dvou sloupcích po 8 bajtech a ve třetím sloupci ASCII znak odpovídající hodnotě. Prvních 54 bajtů odpovídá hlavičce paketu, následují data protokolu Kerberos. Jak je možno vidět, jediné tisknutelné znaky jsou ty, které určují jméno klienta, jméno domény a žádané služby.

Nyní je dobré připomenout, jaká data by měl plugin ze zpráv protokolu Kerberos extrahovat. Způsob detekce, tak jak byl popsán v kapitole 4, se zakládá na informacích o jaké tikety bylo kdy požádáno, a jaké tikety byly kdy vydány. Exportér, pro který byl plugin vyvíjen, automaticky značkuje všechny své výstupy časovou značkou, tudíž časový údaj je zajištěn. Ostatní potřebné údaje, které již musí obstarávat plugin, jsou jméno služby, pro kterou je tiket vystaven, a jméno uživatele, kterému tiket náleží. Pro ještě lepší identifikaci zpráv a větší přehlednost byla zvolena možnost údaje doplnit o typ zprávy, kterou bylo o tiket žádáno. Například o TGT je vždy žádáno prostřednictvím AS_REQ zprávy, což může následné zpracování extrahovaných informací velmi ulehčit.

Postup fungování pluginu je tedy následující:

1. Plugin má zaregistrován `update hook` i `add hook`, které jsou oba ovšem obsluhovány stejnou funkcí. Začíná zpracování zprávy.

2. Probíhají kontroly, zda již není možné pro plugin, aby se daného síťového toku vzdal. Toto nastane v případě, kdy použitý port na cílovém stroji (serveru) není 88, tedy port na který bývá nastavován protokol Kerberos [15], dále pokud již v toku byly nalezeny všechny požadované informace, anebo pokud již v daném toku proteklo větší množství paketů, a tedy se již nedá předpokládat přítomnost paketu se zprávou autentizačního protokolu v dalším pokračování toku.

3. Následuje zpracování typu zprávy, které je ve zprávě vždy na pevné pozici.

4. Nakonec probíhá zpracování jména klienta, pro kterého je tiket vystavován, a služby, které tiket náleží. Tyto dvě hodnoty nemají ve zprávě pevnou pozici, proto je nutné po zpracování typu zbytek zprávy systematicky prohledat.

5. Po skončení toku jsou volány `postprocess hook` a `split hook`, které vyplní nevalidní hodnotou všechny nezpracované atributy z důvodu vnitřních principů exportéru a následně rozdělí informace dle směru toku.

Implementace byla provedena následovně. Byla vytvořena privátní struktura pluginu `kerberos_private_t` (výpis z kódu 4) pro ukládání předávaných atributů z pluginu dále. Funkce, která zpracovává `add hook` a `update hook`, byla nazvána `process_kerberos_update` (výpis z kódu 5). Ta je volána exportérem a je ji předán mimo jiné ukazatel na strukturu reprezentující zachycený paket (`flow_packet_t *packet`) a na záznam o toku, který je stejný pro všechny pakety v toku a udržují se v něm informace o daném toku(`flow_record_t *record`).

```
1  typedef struct {
2      size_t data_offset;
3      int plugin_id;
4      int queue_id;
5
6      /* Kerberos attributes*/
7      flow_record_getter_t *c_name_string;
8      flow_record_getter_t *s_name_string;
9      flow_record_getter_t *ticket_request_type;
10     flow_record_getter_t *ticket_response_type;
11 }kerberos_private_t;
```

Listing 4: Privátní struktura pluginu

```
1  static void process_kerberos_update(void *queue_data, flow_packet_t
   ↪  *packet, flow_record_t *record, void *packet_data, void
   ↪  *record_data);
```

Listing 5: Předpis funkce pro zpracování hooks

Ve struktuře reprezentující paket jsou uloženy kompletní data paketu a informace, na jakém bajtu začíná obsah paketu (`payload offset`). Pro další používání je ve funkci uložen pouze obsah, který je získán přičtením offsetu k adrese dat paketu (výpis z kódu 6).

```
1  const unsigned char *payload = packet->data + packet->payload_offset;
```

Listing 6: Posun ukazatele na data obsahu paketu

Následně ve funkci proběhne rozhodování, zda data z paketu dále zpracovávat. Zprávy protokolu Kerberos mohou být zasílány pomocí transportních protokolů jak TCP tak UDP. Pokud byl pro zaslání použit protokol TCP, je na začátek zprávy přidán 4 bajtová hodnota, ve které je uložena délka zprávy (Record Mark) [15]. Pokud tedy je zjištěn protokol TCP, je nejdříve tato hodnota přeskočena přičtení hodnoty 4 k adrese ukazatele na začátek dat paketu. V prvním bajtu, v bitech 4 až 8, je uložen typ zprávy. Pokud je hodnota získaná

z tohoto bajtu rovna některé z hodnot odpovídající zprávám protokolu Kerberos, jsou data dále zpracovávána.

```
1 static void parse_kerberos_string(kerberos_private_t *private,
  ↪ flow_record_t *record, flow_packet_t *packet, plugin_record_t *rd,
  ↪ const unsigned char *payload, const int payload_len) {
```

Listing 7: Předpis funkce pro zpracování zprávy protokolu Kerberos

Hlavní funkce, která zpracovává data z paketu, byla nazvána `parse_kerberos_string` (výpis z kódu 7). Ještě před jejím zavoláním je ovšem v programu volána funkce `parse_ticket_type` (výpis z kódu 8), která nastaví hodnotu předávaného atributu typ zprávy.

V hlavní funkci následně probíhá hledání struktur ve zprávě, ve kterých jsou zapouzdřena jména uživatele a serveru. Toto probíhá systematickým prohledáním celé zprávy. Struktura obsahující požadované informace (definovaný typ informace je *KerberosString* [15]), má definovaný typ *Universal Sequence*. Takovýchto sekvencí se ve zprávě může objevit několik, funkce tedy každou prozkoumá a pokud zjistí, že obsahuje požadovanou informaci, zavolá další funkci `parse_name` (výpis z kódu 8).

```
1     static int parse_ticket_type(flow_record_getter_t *ticket_request_type,
      ↪ flow_record_getter_t *ticket_response_type, flow_record_t *record,
      ↪ const unsigned char *payload, plugin_record_t *rd)
2
3     static int parse_name(flow_record_getter_t *name, flow_record_t
      ↪ *record, const unsigned char *payload, int lenght)
```

Listing 8: Předpisy funkcí

Kontrola, zda je obsažena požadovaná informace, se provádí porovnáním hodnoty jednoho bajtu, který definuje typ obsaženého elementu. Informace vyhledávané pluginem jsou označeny následujícími třemi hodnotami:

1. KRB5-NT-PRINCIPAL (hodnota 1), určuje jméno klienta
2. KRB5-NT-SRV-INST (hodnota 2), určuje jméno služby
3. KRB5-NT-SRV-HST (hodnota 3), určuje jméno služby

Podle nich jde také poznat, zda je element jméno klienta, nebo serveru.

Ve funkci je tedy implementován cyklus, který postupně inkrementuje hodnotu ukazatele a vždy testuje následující bajt. V případě pozitivního nálezu sekvence je testován bajt obsahující informaci o obsahu sekvence. Bylo zjištěno, že ve zprávě protokolu Kerberos typu TGS_REQ není obsaženo jméno klienta a může být obsaženo více elementů se jmény serveru, takže na rozdíl od jiných typů zpráv, kdy cyklus končí nalezením dvojice jmen, při zpracování tohoto typu cyklus neskončí, dokud nedosáhne konce zprávy.

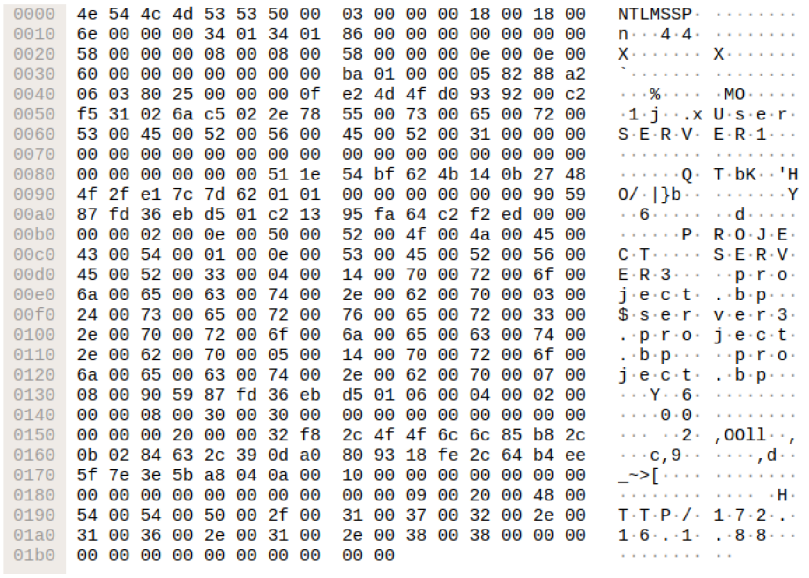
Funkce `parse_ticket_type` nastaví předávaný atribut dle typu zpracovávané zprávy. Pokud jde o zprávy AS_REQ a TGS_REQ nastavuje atribut `ticket_request_type`, pokud o AS_REP a TGS_REP, nastavuje atribut `ticket_response_type`.

Průběh funkce `parse_name` je založen na cyklickém procházení elementu obsahující požadované jméno. Tento element může obsahovat více řetězců, ze kterých se jméno skládá, tudíž funkce je postupně prochází a sestavuje z nich výsledný řetězec, který poté předává funkci pro nastavení atributu.

Pro předání atributu je využita funkce exportéru `flowmon_getter_set`.

Plugin pro NTLM

Pro protokol NTLM byl nalezen manuál popisující velmi přehledně skladbu zprávy protokolu. Tento manuál se nachází na stránce `sourceforge.net` [13]. Zpráva protokolu NTLM, jak byla odchycena při analýze, je zasílána jako součást zprávy protokolu HTTP, ale v rámci této zprávy je zakódována v Base64 [17]. Před zpracováním je tedy nutné dekódování. Protokol NTLM však nebývá zapouzdřen výlučně jen ve zprávě protokolu HTTP. Pro tyto případy je v pluginu přeskočeno dekódování a je přistoupeno rovnou k prohledávání.



Obrázek 5.2: Dekódovaná zpráva NTLM

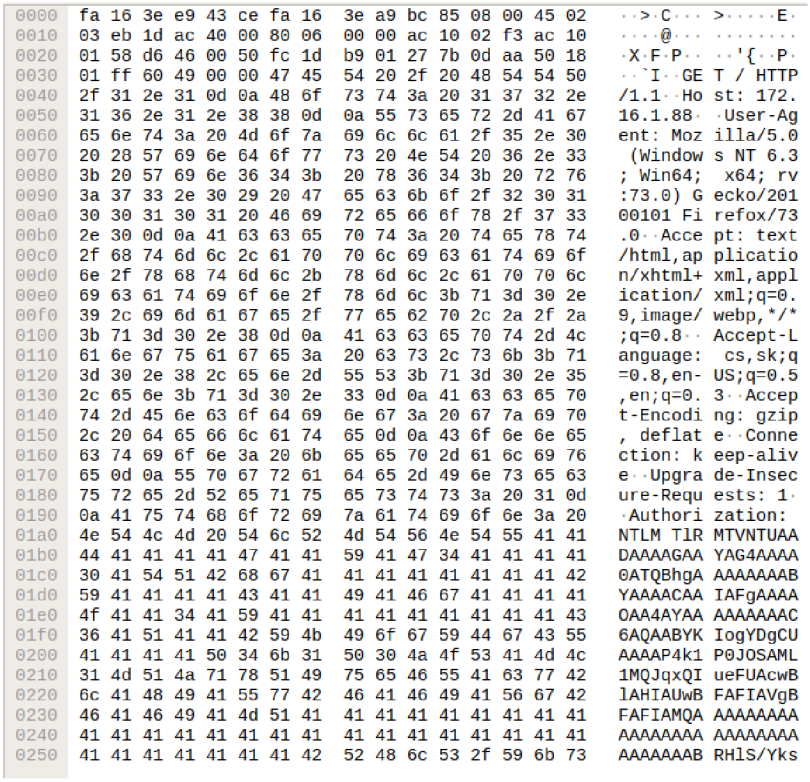
Na obrázku 5.2 je vidět příklad dekódované zprávy NTLM v program Wireshark. Opět vidíme, že tisknutelné znaky jsou jméno klienta a jméno serveru, což je v pořádku, protože přesně tyto dvě informace je potřeba ze zprávy extrahovat, jak bylo popsáno v kapitole 4.

Jak je vidět na obrázku 5.3, který obsahuje část paketu s NTLM zprávou, jak byla přijata, sekce pro NTLM ve zprávě protokolu HTTP je uvedena předpisem '*Authorization:* '. Následuje řetězec zakódovaného textu, který jediný je pro potřeby pluginu zajímavý.

Postup fungování pluginu je následující:

1. Plugin má zaregistrován `update hook` i `add hook`, pro každý z nich je implementována příslušná funkce. Je kontrolován protokol vrstvy L4.

2. Probíhají kontroly, zda již není možné pro plugin, aby se daného síťového toku vzdal.Toto nastane v případě, kdy již v toku byly nalezeny všechny požadované informace, anebo pokud již v daném toku proteklo větší množství paketů, a tedy již se nedá předpokládat přítomnost paketu se zprávou autentizačního protokolu v dalším pokračování toku.



Obrázek 5.3: Zpráva NTLM, jak byla přijata ze síťového toku

3. V případě, že předchozí kontroly jsou úspěšné, je vyhledána ve zprávě sekce obsahující NTLM řetězec, který je v případě HTTP zprávy dekódován.

4. Jako první je kontrolován typ zprávy, který je identifikován hodnotou jednoho bajtu. Zajímavý pro plugin je pouze typ zprávy 3, protože pomocí tohoto typu zprávy probíhá autentizace.

5. Následně jsou zpracovány jména klienta, který se autentizuje, a serveru, ze kterého autentizace probíhá. Informace kde ve zprávě se tyto dvě hodnoty nachází, jsou obsaženy v takzvaných `security bufferech`, které se nachází na přesně dané pozici na začátku zprávy.

Implementace byla provedena následovně. Byla vytvořena privátní struktura pluginu `ntlm_private_t` (výpis z kódu 9) pro ukládání předávaných atributů z pluginu dále.

Funkce, která zpracovává `add hook` byla nazvána `process_ntlm_add()` a v rámci jejího průběhu byla nastavena kontrola protokolu vrstvy L4, který musí být TCP (výpis z kódu 10).

Funkce, která zpracovává `update hook` byla nazvána `process_ntlm_update()` (Výpis z kódu 11). Obě funkce byly vytvořeny ve stejném stylu jako byla vytvořena podobná funkce u pluginu po protokol Kerberos.

Ve funkci `process_ntlm_update()` probíhá, za předpokladu že je zjištěna HTTP zpráva, vyhledání předpisu identifikujícího část s řetězcem NTLM (’*Authorization:* ’) pomocí funkce `memmem()` (výpis z kódu 12). V opačném případě je rovnou vyhledáván předpis ’*NTLMSSP*’.

```
typedef struct {
    int queue_id;

    /* NTLM attributes*/
    flow_record_getter_t *username_string;
    flow_record_getter_t *s_name_string;
}ntlm_private_t;
```

Listing 9: Privátní struktura pluginu

```
static void process_ntlm_add(void *queue_data, flow_packet_t * packet,
↪  flow_record_t * record, void *packet_data, void *record_data){

    if ((packet->record.l4_protocol != IPPROTO_TCP)) {
        ntlm_private_t *private = queue_data;
        flowmon_hook_update_remove(record, private->queue_id,
        ↪  process_ntlm_update);
    }

}
```

Listing 10: Funkce zpracovávající add hook

```
static void process_ntlm_update(void *queue_data, flow_packet_t
↪  *packet, flow_record_t *record, void *packet_data, void
↪  *record_data){
```

Listing 11: Předpis funkce zpracovávající update hook

```
char *authorization = "Authorization: ";

char* authenticate_string = NULL;
authenticate_string = memmem(payload, payload_len, authorization,
↪  strlen(authorization));
```

Listing 12: Vyhledání NTLM zprávy v HTTP zprávě

U protokolu HTTP ve funkci následuje kontrola, zda jde o autentizaci pomocí NTLM, což je provedeno kontrolou, zda první znaky následující po předpisu jsou rovny řetězci "NTLM", případně "Negotiate". V případě úspěchu je provedeno dekódování. K tomu byla využita knihovna pro práci s kódováním Base64 volně dostupná na internetu pod licencí MIT od autora Joe DF (joedf@ahkscript.org). Využita byla funkce `b64_decode` (výpis z kódu 14). Následně je nalezený řetězec předán ke zpracování. Hlavní funkce, která zpracovává data, byla nazvána `parse_ntlm_string()` (výpis z kódu 13).

```
1    static void parse_ntlm_string(ntlm_private_t *private, flow_record_t
     ↪  *record, flow_packet_t *packet, plugin_record_t *rd, const unsigned
     ↪  char *payload, const uint16_t payload_len) {
```

Listing 13: Předpis funkce zpracovávající zprávu protokolu NTLM

```
1    int decoded_data_len =  b64_decode(data, data_len, decoded_data);
```

Listing 14: Volání funkce pro dekódování zprávy protokolu NTLM

Z této funkce jsou volány další funkce, pro kontrolu a zpracování atributů předávaných pluginem dále. Funkce jsou následující.

1. `int parse_username()`
2. `int parse_s_name()`

Funkcím je předán ukazatel na začátek `security bufferu`, který obsahuje informace o zpracovávaném údaji a atribut z privátní struktury pluginu.

Dříve, než začne zpracování zprávy, je provedena kontrola na typ zprávy. Pokud toto číslo není rovno 3, zpracovávání zprávy se ukončí.

Funkce `parse_username()` `parse_s_name()` jsou prakticky identické. Struktura bufferu, který obsahuje informace o potřebném údaji, je následující:

- `short` (2B), délka údaje
- `short` (2B), maximální délka údaje
- `long` (4B), posun pozice údaje od začátku zprávy (offset)

Dle informací získaných z bufferu je tedy určeno, kde se nachází požadovaný řetězec o dané délce. Zde je nutné upozornit na skutečnost, kterou lze vyčíst i z obrázku 5.2. Údaj je ve zprávě uložen ne jako klasický řetězec, ale text je v kódování UTF-16, tedy na jeden znak připadají dva bajty. Logiku vyhledání údaje ve zprávě je možno vidět na výpisu z kódu 15. Atributy jsou následně předány stejným způsobem, jako u pluginu pro protokol Kerberos.

```
    memcpy(&lenght, buffer, sizeof(uint16_t));
    memcpy(&offset, buffer + NTLM_LONG, sizeof(uint32_t));

    char servername[lenght];

    memset(servername, '\0', sizeof(char));

    const unsigned char *name = payload + offset;

    memcpy(servername, name, lenght);
```

Listing 15: Volání funkce pro dekódování zprávy protokolu NTLM

5.2 Testování

Pro účely testování byl firmou Flowmon zpřístupněn obraz operačního systému CentOS, spustitelný ve virtuálním prostředí programem VMware, kde byl nainstalovaný a spustitelný exportér. Na tomto stroji se také nacházely všechny knihovny potřebné pro překlad a kompilaci pluginu.

Plugin je exportérem využíván ve formě sdílené knihovny (formát souboru s příponou .so). Využitý Makefile pro překlad protokolu NTLM, který byl upraven z toho dodaného firmou Flowmon k překladu pluginu zpracovávajícího protokol SSH, je možno vidět na výpisu z kódu 16.

```
    OBJS1 = plugin_process_ntlm.o base64.o
    PLUGIN1 = plugin-process-ntlm

    CFLAGS = -Wall -Wextra -DFLOWMON_PLUGIN -O2 -fPIC --std=gnu99 -I
    ↪  /usr/include/flowmonexp5
    LDFLAGS =  -lm
    DEPS = base64.h

    all: $(PLUGIN1).so

    $(PLUGIN1).so: $(OBJS1)
            gcc -shared -o $(PLUGIN1).so $(LDFLAGS) $(OBJS1) -lrt
            ↪  /usr/lib64/libflowmonexp5.so

    clean:
            -rm $(OBJS1) $(PLUGIN1).so
```

Listing 16: Makefile

Exportér je spouštěn příkazem `flowmonexp5` z příkazové řádky. Zároveň se spuštěním, je nutné jako argument uvést nastavení exportéru, se kterým je spouštěn. K tomu může být využit soubor ve formátu `json`, jehož název je uveden jako argument, a který obsahuje nastavení exportéru a seznam pluginů, se kterými je exportér spouštěn. Ukázkový výpis

ze souboru je možno najít v přílohách. Exportér vypisuje výstup na `stdout`, který může být přesměrován do souboru. Pro účely testování byl tedy exportér spouštěn následujícím příkazem:

```
flowmonexp5 [filename].json > output
```

Exportér má několik možností, z jakých zdrojů může brát data. Pro účely testování byly využity jak pcap soubory, které byly odchycené v rámci této práce ve fázi analýzy, tak ty dostupné z internetu. Aby exportér bral data z těchto souborů, bylo potřeba v souboru s nastavením exportéru uvést u vstupního pluginu parametr `pcap-replay` a mimo jiné specifikovat cestu k pcap souboru. Stejně tak výstup exportéru je možno specifikovat, pro účely testování byl zvolen výstup ve formátu json.

Pro účely testování byl vždy exportér spuštěn se specifikovaným jedním souborem, který obsahoval odchycená data ze sítě, obsahující zprávy zkoumaných protokolů. Ukázkový výstup exportéru pro jeden tok je možno vidět na následujícím výpisu.


```
{"BYTES":521,
"BYTES_A":521,
"CRC":12581687515206398025,
"END_NSEC":"2020-05-14 23:28:56.872730216",
"END_NSEC_A":"2020-05-14 23:28:56.872730216",
"FLAG_FLUSH":7,
"FLAG_MISC":1792,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-14 23:28:56.796349227",
"INPUT_INTERFACE":8,
"KERBEROS_CLIENT_NAME":"SERVER2$",
"KERBEROS_SERVER_NAME":"krbtgt PROJECT.BP",
"KERBEROS_TICKET_REQUEST_TYPE":10,
"KERBEROS_TICKET_RESPONSE_TYPE":0,
"L3_IPV4_DST":"172.16.2.243",
"L3_IPV4_SRC":"172.16.2.99",
"L3_PROTO":4,
"L4_PORT_DST":88,
"L4_PORT_SRC":49865,
"L4_PROTO":6,
"L4_TCP_FLAGS":"CE-AP-SF",
"L4_TCP_FLAGS_A":"CE-AP-SF",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":5,
"PACKETS_A":5,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-14 23:28:56.870492920",
"START_NSEC_A":"2020-05-14 23:28:56.870492920"}
```


Na výpisu je vidět informace o toku, včetně informací, které dodal plugin zpracovávající protokol Kerberos. Je vidět, že jde o AS_REQ zprávu (odpovídá hodnotě 10), kterou zaslal

SERVER 2 na službu *krbtgt*. Na dalším výpisu je možno vidět obdobný výpis pro protokol NTLM. Zde se ovšem vyskytl problém s vypisováním řetězce v kódování UTF 16, tudíž bylo vypsáno pouze první písmeno (další přečtený bajt byl totiž nulový), ovšem dle instrukcí firmy Flowmon je žádoucí předávání v tomto tvaru, protože v programech, které zpracovávají data z exportéru je možná konverze na jiné kódování.


```
{"BYTES":899,
"BYTES_A":899,
"CRC":10772518155177636344,
"END_NSEC":"2020-05-18 19:47:13.273508299",
"END_NSEC_A":"2020-05-18 19:47:13.273508299",
"FLAG_FLUSH":7,
"FLAG_MISC":3840,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-18 19:47:13.246963376",
"INPUT_INTERFACE":8,
"L3_IPV4_DST":"172.16.2.243",
"L3_IPV4_SRC":"172.16.1.88",
"L3_PROTO":4,
"L4_PORT_DST":54854,
"L4_PORT_SRC":80,
"L4_PROTO":6,
"L4_TCP_FLAGS":"-E-AP-S-",
"L4_TCP_FLAGS_A":"-E-AP-S-",
"NTLM_SERVER_NAME":"S",
"NTLM_USERNAME":"U",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":2,
"PACKETS_A":2,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-18 19:47:13.271443754",
"START_NSEC_A":"2020-05-18 19:47:13.271443754"}
```


Všechny ostatní testovací vstupy a výstupy je možno najít v přílohách. Soubory pcap používané pro testování, byly za prvé ty odchycené ze sítě ve fázi analýzy a druhé soubory pcap získané z internetu ze stránky `pcapr.net`.

Bohužel nebylo možné vytvořené pluginy otestovat v reálném provozu v nějaké nesimulované síti. Zadávající firma Flowmon neměla prostředky pro to zajistit nějaké testovací prostředí, kde by byl implementován jak protokol Kerberos, tak NTLM, ale které by zároveň nebylo simulované. Příkladem simulovaného prostředí může být síťová struktura vytvořená pro potřeby analýzy této práce, kde prakticky jiný provoz než ten uměle vytvořený a potřebný není. Testování v takové síti je v principu ekvivalentní testování pomocí pcap souborů.

Kapitola 6

Závěr

Cílem této práce bylo popsat a následně analyzovat síťové autentizační protokoly Kerberos, NTLM a SAML 2.0 a pomocí dosažených výsledků vytvořit plugin pro exportér Flowmon sondy zpracovávající tyto protokoly. Analýza měla být provedena především nad síťovým provozem s přítomností zpráv těchto protokolů, a to jak z teoretického, tak praktického hlediska. Hlavním cílem analýzy mělo být vytvoření návrhu, jak detekovat ze síťového provozu možné útoky, které by cílily na tyto protokoly.

V teoretické části analýzy byl blíže zkoumán a popsán princip, jaký používají všechny protokoly pro autentizaci. Následně byly zkoumány a popsány známé útoky, které cílí na obejití nebo prolomení těchto protokolů. Byly blíže zkoumány principy, na kterých tyto útoky zakládají, možné chyby protokolů, které mohou být útoky využity a možnosti, které potenciální útočník může využít, pokud chce prolomit ochranu poskytnutou zkoumanými protokoly.

V praktické části analýzy byla uměle vytvořena síťová struktura, ve které byly následně prováděny dva typy útoků – Pass the Ticket a Pass the Hash. Ve stejné době byl ze sítě odchytáván provoz, který byl podroben velmi blízkému zkoumání za účelem zjištění možností detekce útoků ze síťového toku. Pro každý protokol, za použití informací ze síťového toku i jiných zdrojů, byly tyto možnosti zjištěny. Tato a všechny další části práce kvůli technickým možnostem zpracovávaly pouze protokoly Kerberos a NTLM.

Výsledky dosažené ve fázi analýzy byly poté využity pro vytvoření návrhu způsobů, jak pomocí sledování síťového toku automaticky detekovat popsané útoky. Tyto způsoby zakládají na principu hlídání všech autentizačních zpráv v síti a kontrole, provedené na základě informací dostupných ze zpráv, zda jsou tyto zprávy očekávané. Podle toho byly vytvořeny pluginy pro exportér Flowmon sondy, které po zpracování síťového toku obsahujícího zprávy zkoumaných protokolů vytváří výstup s informacemi, které jsou potřebné pro detekci útoků tak, jak bylo navrženo.

Na závěr bylo provedeno testování těchto pluginů. Pro toto byl využit virtuální obraz systému s nainstalovaným exportérem, kterému byly předkládány soubory se síťovým tokem, obsahujícím zprávy protokolů. Proběhla neautomatizovaná kontrola, zda pluginy vytváří výstup, který obsahuje všechny očekávané kompletní informace, což bylo potvrzeno. Tímto bylo, alespoň pro protokoly Kerberos a NTLM, dosaženo všech cílů práce.

Všechny způsoby detekce útoků, které byly v práci navrženy, nejsou však stoprocentně spolehlivé. Vzhledem k povaze protokolů, založené na zasílání šifrovaných zpráv, je těžké sledovat kompletní proces autentizace, a proto jsou způsoby detekce navržené v této práci spíše způsoby jak upozornit s největší pravděpodobností na možný probíhající útok.

Při interpretaci a použití dosažených výsledků, je nutné si také uvědomit, že síť ve které probíhala analýza, byla uměle vytvořena. Sice byla vytvořena, aby se co nejvíce podobala reálně používané síti, nicméně reálně používanou sítí nebyla. Všechna možná nastavení, která byla v síti provedena, mohla ovlivnit dosažené výsledky, stejně jako všechna nastavení, která provedena nebyla a přitom v jiné síti můžou být standardem, mohla zamezit provedení útoku již v zárodku, případně znemožnit jeho detekci.

Jedním z dalších pokračování práce by mohlo být pokrytí většího množství případů, kdy jsou protokoly využívány. V této práci byl zkoumán, oproti všem možnostem, které můžou protokoly nabízet, pouze celkem úzký výsek případů užití. Pro protokol Kerberos by bylo možné rozšířit možnosti detekce i o další zprávy protokolu, než ty zachytávané pluginem vytvořeným v rámci této práce. Bylo by možné například hlídat využívání všech tiketů v síti. Pro protokol NTLM by bylo možné prozkoumat další případy, kdy se protokol v rámci sítě využívat a implementovat prozkoumávání dalších protokolů, ve kterých může být autentizační zpráva NTLM zapouzdřena.

Další možné pokračování by mohlo být zpracování více útoků. Útoky zpracované v této práci byly zvoleny kvůli jejich popularitě a zajímavosti, nicméně to neznamená, že jiné útoky neexistují a nejsou používány. Pokračování práce by mohlo tyto útoky blíže prozkoumat a následně navrhnout způsob jejich detekce.

Jiné pokračování práce, které se přímo nabízí, je praktická analýza protokolu SAML. Nicméně protokol SAML má velmi velké spektrum možností, kde může být využit, a skladba jeho zpráv se může velmi lišit, protože záleží na každé aplikaci zvlášť, tudíž jeho prozkoumání a zpracování by mohlo posloužit jako téma pro další celou práci.

Literatura

[1] Barker, K. *Cyberattack*. School of Public Policy, University of Calgary, 2019. SPP Briefing Paper. Dostupné z: https://deslibris.ca/ID/10100995.

[2] Bačkor, M. *Monitorovanie sieťových tokov protokolu Kerberos*. Brno, 2015. Bakalářská práce. Masarykova Univerzita, Fakulta Informatiky.

[3] Bellovin, S. M. a Merritt, M. Limitations of the Kerberos Authentication System. *SIGCOMM Comput. Commun. Rev.* New York, NY, USA: Association for Computing Machinery. říjen 1990, roč. 20, č. 5, s. 119–132. Dostupné z: https://doi.org/10.1145/381906.381946. ISSN 0146-4833.

[4] Bouška, P. Active Directory komponenty - domain, tree, forest, site. *Samuraj* [online]. 2008 [cit. 29. prosince 2019]. Dostupné z: https://www.samuraj-cz.com/clanek/active-directory-komponenty-domain-tree-forest-site/.

[5] Broeckelmann, R. Kerberos Wireshark Captures: A Windows Login Example. *Medium* [online]. 2018 [cit. 4. března 2020]. Dostupné z: https://medium.com/@robert.broeckelmann/kerberos-wireshark-captures-a-windows-login-example-151fabf3375a.

[6] Casas, P., Mazel, J. a Owezarski, P. Knowledge-independent traffic monitoring: Unsupervised detection of network attacks. *IEEE Network*. IEEE. 2012, roč. 26, č. 1, s. 13–21. ISSN 1558-156X.

[7] Duncan, R. An Overview of Different Authentication Methods and Protocols. *Information Security Reading Room*. SANS Institute. [online]. 2001. Published as part of SANS Institute graduation program. Dostupné z: https://www.sans.org/reading-room/whitepapers/authentication/overview-authentication-methods-protocols-118.

[8] Fiu, S. Global market share held by operating systems for desktop PCs, from January 2013 to January 2020. *Statista* [online]. 2020 [cit. 2. května 2020]. Dostupné z: https://www.statista.com/statistics/218089/global-market-share-of-windows-7/.

[9] Flowmon. *Flowmon exporter documentation*. Flowmon.

[10] Garman, J. *Kerberos : definitive guide*. Vyd. 1. New York: O´Reilly, 2003. ISBN 0-596-00403-6.

[11] gentilkiwi. *Mimikatz* [online]. Volně dostupný program. Dostupné z: https://github.com/gentilkiwi/mimikatz.

[12] GhostPack. *Rubeus* [online]. Volně dostupný program. Dostupné z: https://github.com/GhostPack/Rubeus.

[13] Glass, E. *The NTLM Authentication Protocol and Security Support Provider* [online]. SourceForge, 2006 [cit. 26. dubna 2020]. Dostupné z: http://davenport.sourceforge.net/ntlm.html.

[14] Group, N. W. *Advanced Encryption Standard (AES) Encryption for Kerberos 5* [online]. 1. vyd. únor 2005.

[15] Group, N. W. *The Kerberos Network Authentication Service (V5)* [online]. 1. vyd. červenec 2005.

[16] Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R. et al. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE COMMUNICATIONS SURVEYS AND TUTORIALS.* [online]. 2014, roč. 16, č. 4. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6814316. ISSN 1553-877X.

[17] (IETF), I. E. T. F. *The Base16, Base32, and Base64 Data Encodings* [online]. říjen 2006.

[18] (IETF), I. E. T. F. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* [online]. 1. vyd. Září 2013.

[19] Januszkiewicz, P. Pass-The-Hash Attack Tutorial. *CQURE Academy* [online]. 2016 [cit. 2. března 2020]. Dostupné z: https://cqureacademy.com/blog/identity-theft-protection/pass-hash-attack-tutorial.

[20] Jensen, J. Attacking SSO: Common SAML Vulnerabilities and Ways to Find Them. *NETSPI* [online]. 2017 [cit. 2. května 2019]. Dostupné z: https://blog.netspi.com/attacking-sso-common-saml-vulnerabilities-ways-find/.

[21] Microsoft. *PsTools* [online]. Volně dostupná knihovna nástrojů pro Microsoft Windows. Dostupné z: https://download.sysinternals.com/files/PSTools.zip.

[22] Microsoft. *Active Directory Domain Services* [online]. Microsoft Corporation, leden 2013 [cit. 29. prosince 2019]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc753910(v=ws.10).

[23] Microsoft. *Kerberos Authentication Overview* [online]. Microsoft Corporation, říjen 2016 [cit. 2. května 2020]. Dostupné z: https://docs.microsoft.com/en-us/windows-server/security/kerberos/kerberos-authentication-overview.

[24] Microsoft. *Microsoft NTLM* [online]. Microsoft Corporation, květen 2018 [cit. 26. listopadu 2019]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/secauthn/microsoft-ntlm.

[25] Microsoft. *Konfigurace nastavení poskytovatele SAML 2.0 pro portály* [online]. Microsoft Corporation, říjen 2019 [cit. 2. května 2020]. Dostupné z: https://docs.microsoft.com/cs-cz/powerapps/maker/portals/configure/configure-saml2-settings.

[26] Microsoft. *NTLM v2 Authentication* [online]. Microsoft Corporation, září 2019 [cit. 26. listopadu 2019]. Dostupné z: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nlmp/5e550938-91d4-459f-b67d-75d70009e3f3.

[27] Microsoft. *Security Considerations for Implementers* [online]. Microsoft Corporation, září 2019 [cit. 29. prosince 2019]. Dostupné z: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nlmp/1e846608-4c5f-41f4-8454-1b91af8a755b.

[28] MIT. *Kerberos: The Network Authentication Protocol* [online]. 2020. Aktualizováno 27. 4. 2020 [cit. 2. května 2019]. Dostupné z: https://web.mit.edu/kerberos/.

[29] MIT. *MIT Kerberos Documentation* [online]. MIT, únor 2020 [cit. 2. května 2020]. Dostupné z: https://web.mit.edu/kerberos/krb5-latest/doc/index.html.

[30] OASIS. *Security Assertion Markup Language(SAML) V2.0 Technical Overview* [online]. Organization for the Advancement of Structured Information Standards, březen 2008 [cit. 2. května 2020]. Dostupné z: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html.

[31] OneLogin. How does single sign-on work? *OneLogin* [online]. 2020 [cit. 2. května 2019]. Dostupné z: https://www.onelogin.com/learn/how-single-sign-on-works.

[32] Panayi, C. Passing-the-Hash to NTLM Authenticated Web Applications. *F-Secure Labs* [online]. 2018 [cit. 2. března 2020]. Dostupné z: https://labs.f-secure.com/blog/pth-attacks-against-ntlm-authenticated-web-applications/.

[33] Pérez, E. Kerberos (II): How to attack Kerberos? *Tarlogic* [online]. 2019 [cit. 20. března 2020]. Dostupné z: https://www.tarlogic.com/en/blog/how-to-attack-kerberos/.

[34] Reiner, S. Golden SAML: Newly Discovered Attack Technique Forges Authentication to Cloud Apps. *CYBERARK* [online]. 2017 [cit. 2. května 2020]. Dostupné z: https://www.cyberark.com/threat-research-blog/golden-saml-newly-discovered-attack-technique-forges-authentication-cloud-apps/.

[35] Sanders, C. Disecting the Pass the Hash Attack. *TechGenix* [online]. 2010 [cit. 26. prosince 2019]. Dostupné z: http://techgenix.com/dissecting-pass-hash-attack/.

[36] Stanek, W. R. W. R. *Active Directory : kapesní rádce administrátora.* Vyd. 1. Brno: Computer Press, 2009. ISBN 978-80-251-2555-7.

[37] Steiner, J. G., Neuman, C. a Schiller, J. I. Kerberos: An Authentication Service for Open Network Systems. *Project Athena, MIT.* MIT. [online]. 1988, č. 1. Dostupné z: https://www3.nd.edu/~dthain/courses/cse66771/summer2014/papers/kerberos.pdf.

[38] team, W. *GitHub Wireshark* [online]. 2020 [cit. 5. července 2020]. Dostupné z: https://github.com/boundary/wireshark.

[39] Union, I. T. *X.690 : Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)* [online]. 1. vyd. Srpen 2015.

[40] Warren, J. How to Detect Pass-the-Ticket Attacks. *Insider Threat Security Blog* [online]. 2019 [cit. 11. ledna 2020]. Dostupné z: https://blog.stealthbits.com/detect-pass-the-ticket-attacks.

[41] Warren, J. How to Detect Pass-the-Hash Attacks. *Insider Threat Security Blog* [online]. 2019 [cit. 11. ledna 2020]. Dostupné z: https://blog.stealthbits.com/how-to-detect-pass-the-hash-attacks/.

[42] Wikipedia. Kerberos (protocol). *Wikipedia* [online]. 2020. Aktalizovno 28. 4. 2020 [cit. 2. května 2020]. Dostupné z: https://cs.wikipedia.org/wiki/Kerberos_(protokol).

[43] Wireshark. *Display Filter Reference: Kerberos* [online]. Wireshark [cit. 5. února 2020]. Dostupné z: https://www.wireshark.org/docs/dfref/k/kerberos.html#.

Příloha A

Obsah přiloženého média

Přiložené médium obsahuje část se zdrojovými kódy a část se soubory pcap s odchyceným síťovým tokem.

Zdrojové kódy

Tato část obsahuje zdrojové kódy pluginů, vytvořených v této práci a knihoven funkcí, které pluginy využívají.

Seznam souborů:

- plugin-process-kerberos.c - Zrojový soubor pluginu zpracovávajícího protokol Kerberos
- plugin-process-ntlm.c - Zdrojový soubor pluginu zpracovávajícího protokol NTLM
- base64.h - Knihovna funkcí pro práci s kódováním Base64(Převzato z internetu)
- base64.c - Zdrojové kódy funkcí pro práci s kódováním Base64 (Převzato z internetu)
- Makefile - Makefile pro kompilaci pluginu Kerberos, pro kompilaci pro protokol NTLM nutno změnit názvy souborů

Pro kompilaci je nutné mít zdrojové soubory knihoven Flowmon exportéru!!!

Soubory pro analýzu a testování

Tato část obsahuje PCAP soubory, které byly odchyceny ze sítě vystavěné po potřeby vypracování práce.

Soubory jsou rozděleny do dvou skupin, na první, obsahující soubory, kterých probíhala analýza, a druhou, obsahující výňatky z prvních souborů, používané k testování, přičemž každé náleží vlastní složka.

Ve složce se soubory pro analýzu jsou soubory obsahující zprávy Kerberos a NTLM, které byly po síti zasílány při klasickém přihlášení i pří útocích (Pass the Ticket a Pass the Hash).

Ve složce se soubory pro testování vytvořených pluginů jsou za prvé výřezy souborů, které byly zachyceny při analýze a za druhé soubory stáhnuté z internetu, které obsahují zprávy zkoumaných protokolů.

Ke každému testovacímu souboru je připojen json soubor s výstupem exportéru.

Příloha B

Ukázky testovacích výstupů

Kerberos

FlowOne


```
{
"BYTES":521,
"BYTES_A":521,
"CRC":12581687515206398025,
"END_NSEC":"2020-05-14 23:28:56.872730216",
"END_NSEC_A":"2020-05-14 23:28:56.872730216",
"FLAG_FLUSH":7,
"FLAG_MISC":1792,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-14 23:28:56.796349227",
"INPUT_INTERFACE":8,
"KERBEROS_CLIENT_NAME":"SERVER2$",
"KERBEROS_SERVER_NAME":"krbtgt PROJECT.BP",
"KERBEROS_TICKET_REQUEST_TYPE":10,
"KERBEROS_TICKET_RESPONSE_TYPE":0,
"L3_IPV4_DST":"172.16.2.243",
"L3_IPV4_SRC":"172.16.2.99",
"L3_PROTO":4,
"L4_PORT_DST":88,
"L4_PORT_SRC":49865,
"L4_PROTO":6,
"L4_TCP_FLAGS":"CE-AP-SF",
"L4_TCP_FLAGS_A":"CE-AP-SF",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":5,
"PACKETS_A":5,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-14 23:28:56.870492920",
"START_NSEC_A":"2020-05-14 23:28:56.870492920"
}
{
"BYTES":1639,
"BYTES_A":1639,
"CRC":12581687515206398025,
"END_NSEC":"2020-05-14 23:28:56.872730940",
"END_NSEC_A":"2020-05-14 23:28:56.872730940",
"FLAG_FLUSH":7,
"FLAG_MISC":1793,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-14 23:28:56.796349227",
"INPUT_INTERFACE":8,
"KERBEROS_CLIENT_NAME":"SERVER2$",
"KERBEROS_SERVER_NAME":"krbtgt PROJECT.BP",
"KERBEROS_TICKET_REQUEST_TYPE":0,
```

```
"KERBEROS_TICKET_RESPONSE_TYPE":11,
"L3_IPV4_DST":"172.16.2.99",
"L3_IPV4_SRC":"172.16.2.243",
"L3_PROTO":4,
"L4_PORT_DST":49865,
"L4_PORT_SRC":88,
"L4_PROTO":6,
"L4_TCP_FLAGS":"-E-AP-S-",
"L4_TCP_FLAGS_A":"-E-AP-S-",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":3,
"PACKETS_A":3,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-14 23:28:56.872090980",
"START_NSEC_A":"2020-05-14 23:28:56.872090980"
}
```


FlowTwo


```
{
"BYTES":516,
"BYTES_A":516,
"CRC":11418061428709166634,
"END_NSEC":"2020-05-14 23:31:18.102117302",
"END_NSEC_A":"2020-05-14 23:31:18.102117302",
"FLAG_FLUSH":7,
"FLAG_MISC":1792,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-14 23:31:18.022926193",
"INPUT_INTERFACE":8,
"KERBEROS_CLIENT_NAME":"User",
"KERBEROS_SERVER_NAME":"krbtgt PROJECT.BP",
"KERBEROS_TICKET_REQUEST_TYPE":10,
"KERBEROS_TICKET_RESPONSE_TYPE":0,
"L3_IPV4_DST":"172.16.2.243",
"L3_IPV4_SRC":"172.16.2.99",
"L3_PROTO":4,
"L4_PORT_DST":88,
"L4_PORT_SRC":49247,
"L4_PROTO":6,
"L4_TCP_FLAGS":"CE-AP-SF",
"L4_TCP_FLAGS_A":"CE-AP-SF",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":5,
"PACKETS_A":5,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-14 23:31:18.100041743",
"START_NSEC_A":"2020-05-14 23:31:18.100041743"
}
{
"BYTES":1597,
"BYTES_A":1597,
"CRC":11418061428709166634,
"END_NSEC":"2020-05-14 23:31:18.102117908",
"END_NSEC_A":"2020-05-14 23:31:18.102117908",
"FLAG_FLUSH":7,
"FLAG_MISC":1793,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-14 23:31:18.022926193",
"INPUT_INTERFACE":8,
"KERBEROS_CLIENT_NAME":"User",
"KERBEROS_SERVER_NAME":"krbtgt PROJECT.BP",
"KERBEROS_TICKET_REQUEST_TYPE":0,
"KERBEROS_TICKET_RESPONSE_TYPE":11,
"L3_IPV4_DST":"172.16.2.99",
```

```
"L3_IPV4_SRC":"172.16.2.243",
"L3_PROTO":4,
"L4_PORT_DST":49247,
"L4_PORT_SRC":88,
"L4_PROTO":6,
"L4_TCP_FLAGS":"-E-AP-S-",
"L4_TCP_FLAGS_A":"-E-AP-S-",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":3,
"PACKETS_A":3,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-14 23:31:18.101760018",
"START_NSEC_A":"2020-05-14 23:31:18.101760018"
}
```


FlowThree


```
{
"BYTES":1552,
"BYTES_A":1552,
"CRC":14947557962809926617,
"END_NSEC":"2020-05-15 00:33:16.575271174",
"END_NSEC_A":"2020-05-15 00:33:16.575271174",
"FLAG_FLUSH":7,
"FLAG_MISC":1792,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-15 00:33:16.539633468",
"INPUT_INTERFACE":8,
"KERBEROS_CLIENT_NAME":"User",
"KERBEROS_SERVER_NAME":"host server2.project.bp",
"KERBEROS_TICKET_REQUEST_TYPE":12,
"KERBEROS_TICKET_RESPONSE_TYPE":0,
"L3_IPV4_DST":"172.16.2.243",
"L3_IPV4_SRC":"172.16.2.99",
"L3_PROTO":4,
"L4_PORT_DST":88,
"L4_PORT_SRC":49248,
"L4_PROTO":6,
"L4_TCP_FLAGS":"CE-AP-SF",
"L4_TCP_FLAGS_A":"CE-AP-SF",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":4,
"PACKETS_A":4,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-15 00:33:16.571780933",
"START_NSEC_A":"2020-05-15 00:33:16.571780933"
}
{
"BYTES":1535,
"BYTES_A":1535,
"CRC":14947557962809926617,
"END_NSEC":"2020-05-15 00:33:16.575305637",
"END_NSEC_A":"2020-05-15 00:33:16.575305637",
"FLAG_FLUSH":7,
"FLAG_MISC":1793,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-15 00:33:16.539633468",
"INPUT_INTERFACE":8,
"KERBEROS_CLIENT_NAME":"User",
"KERBEROS_SERVER_NAME":"host server2.project.bp",
"KERBEROS_TICKET_REQUEST_TYPE":0,
"KERBEROS_TICKET_RESPONSE_TYPE":13,
"L3_IPV4_DST":"172.16.2.99",
"L3_IPV4_SRC":"172.16.2.243",
"L3_PROTO":4,
```

```
"L4_PORT_DST":49248,
"L4_PORT_SRC":88,
"L4_PROTO":6,
"L4_TCP_FLAGS":"-E-AP-S-",
"L4_TCP_FLAGS_A":"-E-AP-S-",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":3,
"PACKETS_A":3,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-15 00:33:16.574653654",
"START_NSEC_A":"2020-05-15 00:33:16.574653654"
}
```


FlowFour


```
{
"BYTES":1819,
"BYTES_A":1819,
"CRC":18104693562031998135,
"END_NSEC":"2020-05-15 00:35:00.734925293",
"END_NSEC_A":"2020-05-15 00:35:00.734925293",
"FLAG_FLUSH":7,
"FLAG_MISC":3840,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-15 00:35:00.690762690",
"INPUT_INTERFACE":8,
"KERBEROS_CLIENT_NAME":"Administrator",
"KERBEROS_SERVER_NAME":"HTTP server2.project.bp",
"KERBEROS_TICKET_REQUEST_TYPE":12,
"KERBEROS_TICKET_RESPONSE_TYPE":0,
"L3_IPV4_DST":"172.16.3.246",
"L3_IPV4_SRC":"172.16.2.243",
"L3_PROTO":4,
"L4_PORT_DST":49277,
"L4_PORT_SRC":88,
"L4_PROTO":6,
"L4_TCP_FLAGS":"-E-AP-S-",
"L4_TCP_FLAGS_A":"-E-AP-S-",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":4,
"PACKETS_A":4,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-15 00:35:00.713068834",
"START_NSEC_A":"2020-05-15 00:35:00.713068834"
}
{
"BYTES":1834,
"BYTES_A":1834,
"CRC":18104693562031998135,
"END_NSEC":"2020-05-15 00:35:00.734924013",
"END_NSEC_A":"2020-05-15 00:35:00.734924013",
"FLAG_FLUSH":7,
"FLAG_MISC":3841,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-15 00:35:00.690762690",
"INPUT_INTERFACE":8,
"KERBEROS_CLIENT_NAME":"Administrator",
"KERBEROS_SERVER_NAME":"HTTP server2.project.bp",
"KERBEROS_TICKET_REQUEST_TYPE":0,
"KERBEROS_TICKET_RESPONSE_TYPE":13,
"L3_IPV4_DST":"172.16.2.243",
"L3_IPV4_SRC":"172.16.3.246",
"L3_PROTO":4,
"L4_PORT_DST":88,
"L4_PORT_SRC":49277,
```

```
"L4_PROTO":6,
"L4_TCP_FLAGS":"---AP--F",
"L4_TCP_FLAGS_A":"---AP--F",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":5,
"PACKETS_A":5,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-15 00:35:00.728221215",
"START_NSEC_A":"2020-05-15 00:35:00.728221215"
}
```


NTLM

Flow1


```
{
"BYTES":899,
"BYTES_A":899,
"CRC":10772518155177636344,
"END_NSEC":"2020-05-18 19:47:13.273508299",
"END_NSEC_A":"2020-05-18 19:47:13.273508299",
"FLAG_FLUSH":7,
"FLAG_MISC":3840,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-18 19:47:13.246963376",
"INPUT_INTERFACE":8,
"L3_IPV4_DST":"172.16.2.243",
"L3_IPV4_SRC":"172.16.1.88",
"L3_PROTO":4,
"L4_PORT_DST":54854,
"L4_PORT_SRC":80,
"L4_PROTO":6,
"L4_TCP_FLAGS":"-E-AP-S-",
"L4_TCP_FLAGS_A":"-E-AP-S-",
"NTLM_SERVER_NAME":"S",
"NTLM_USERNAME":"U",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":2,
"PACKETS_A":2,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-18 19:47:13.271443754",
"START_NSEC_A":"2020-05-18 19:47:13.271443754"
}
{
"BYTES":1510,
"BYTES_A":1510,
"CRC":10772518155177636344,
"END_NSEC":"2020-05-18 19:47:13.278856416",
"END_NSEC_A":"2020-05-18 19:47:13.278856416",
"FLAG_FLUSH":7,
"FLAG_MISC":3841,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-18 19:47:13.246963376",
"INPUT_INTERFACE":8,
"L3_IPV4_DST":"172.16.1.88",
"L3_IPV4_SRC":"172.16.2.243",
"L3_PROTO":4,
"L4_PORT_DST":80,
"L4_PORT_SRC":54854,
"L4_PROTO":6,
"L4_TCP_FLAGS":"---AP---",
"L4_TCP_FLAGS_A":"---AP---",
"NTLM_SERVER_NAME":"S",
```

```
"NTLM_USERNAME":"U",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":3,
"PACKETS_A":3,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-18 19:47:13.273427468",
"START_NSEC_A":"2020-05-18 19:47:13.273427468"
}
{
"BYTES":40,
"BYTES_A":40,
"CRC":10772518153413160311,
"END_NSEC":"2020-05-18 19:47:13.273481664",
"END_NSEC_A":"2020-05-18 19:47:13.273481664",
"FLAG_FLUSH":7,
"FLAG_MISC":1792,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-18 19:47:13.246963376",
"INPUT_INTERFACE":8,
"L3_IPV4_DST":"172.16.2.243",
"L3_IPV4_SRC":"172.16.1.88",
"L3_PROTO":4,
"L4_PORT_DST":54843,
"L4_PORT_SRC":80,
"L4_PROTO":6,
"L4_TCP_FLAGS":"---A---F",
"L4_TCP_FLAGS_A":"---A---F",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":1,
"PACKETS_A":1,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-18 19:47:13.273481664",
"START_NSEC_A":"2020-05-18 19:47:13.273481664"
}
{
"BYTES":40,
"BYTES_A":40,
"CRC":10772518153413160311,
"END_NSEC":"2020-05-18 19:47:13.273499629",
"END_NSEC_A":"2020-05-18 19:47:13.273499629",
"FLAG_FLUSH":7,
"FLAG_MISC":1793,
"FLOWMON_STARTUP_TIME_NSEC":"2020-05-18 19:47:13.246963376",
"INPUT_INTERFACE":8,
"L3_IPV4_DST":"172.16.1.88",
"L3_IPV4_SRC":"172.16.2.243",
"L3_PROTO":4,
"L4_PORT_DST":80,
"L4_PORT_SRC":54843,
"L4_PROTO":6,
"L4_TCP_FLAGS":"---A----",
"L4_TCP_FLAGS_A":"---A----",
"ONE":1,
"OUTPUT_INTERFACE":0,
"PACKETS":1,
"PACKETS_A":1,
"QUEUE_ID":0,
"SAMPLING_ALGORITHM":0,
"SAMPLING_RATE":0,
"START_NSEC":"2020-05-18 19:47:13.273499629",
"START_NSEC_A":"2020-05-18 19:47:13.273499629"
}
```

Příloha C

ASN.1 and BER Encoding

Převzato z práce Description of IEC 61850Communication, Ing. Petr Matoušek, PhD.

Abstract Syntax Notation 1 (ASN.1, defined by ITU-T X.680) specifies the following categories of data types:

- Primitive data types (universal class 00)
  - BOOLEAN –universal class tag 1
  - INTEGER –universal class tag 2
  - BIT STRINT –universal class tag 3
  - OCTET STRING –universal class tag 4
  - NULL –universal class tag 5
  - OBJECT IDENTIFIER –universalclass tag 6
  - ObjectDescriptor –universal class tag 7
  - EXTERNAL –universal class tag 8
  - REAL –universal class tag 9
  - ENUMERATED –universal class tag 10
  - UTF8String –universal class tag 12
  - NumericString –universal class tag 18
  - PrintableString –universal class tag 19
  - IA5String –universal class tag
  - UTCTime –universal class tag 23
  - GeneralizedTime –universal class tag 24
  - GraphicString –universal class tag 25
  - VisibleString –universal class tag 26
  - GeneralString –universal class tag 27
  - UniversalString –universal class tag 28
  - CHARACTER STRING –universal tag 29
- Application-wide data types (class 01)

  - Not standardized but defined by each application.

- Constructor data types
  - SEQUENCE, SEQUENCE OF –universal class tag 16
  - SET, SET OF –universal class tag 17
  - CHOICE
  - SELECTION
  - ANY

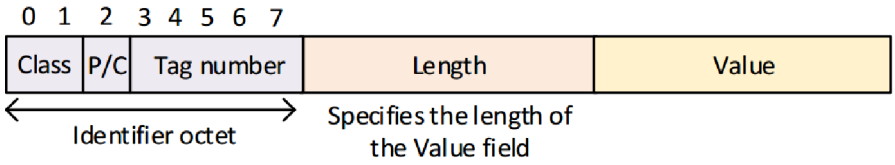
Basic Encoding Rules (BER, standard ITU-T X.690) defines transfer syntax of ASN.1 data structures transmitted between applications. BER describes a method how to encode values of ANS.1 data as a string of octets. It encodes an ASN.1 value as a triplet TLV (type-length-value) that includes an identifier of the data type, length of the value, and the value itself, see the following figure.

- *Type*(or identifier) is one-byte value that indicates the ASN.1 type, see below.
- *Length* indicates the length of the actual value representation.
- *Value* represents the value of ASN.1 type as a string of octets. For constructed types the value can be an embedded TLV triplet.

Example 1: sequence 41 02 3F 22 (hex)

- Type 41 = 0100 0001 (binary) denotes class 01 (application), primitive type (0) and the application tag is 1 (00001), see below. Application tag 1 in SNMP is Counter32 data type.
- 02 gives the length of the data, e.g., 2 bytes.
- 3F 22 is the value of the variable oftype Counter 32. The value is 16,162 in decimal.

The *identifier* specifies the ASN.1 data type, the class of the type, and the method of encoding, see the following figure:



- *The first two bits* determine the class of the ASN.1 data type:
  - Universal (00) -universal data types (primitive or constructive) are given by the standard,
  - Application (01),

  - Context-specific (10),
  - Private (11).

- *The third bit* describes the encoding method: primitive (0) or constructed (1) form.
  - If set to 1, constructed data types as SEQUENCE, SET, CHOICE, etc. are used. It also means that another TLV triplet is embedded as a value.

- *The last five bits* identify the data type. This is called a tag. Universal data type tags are listed above. Application, context-specific and private tags are defined by the application.

The length can be encoding in three forms:

- *short definite length* (primitive form): 1 byte value if the MSB is 0, i.e., for length 0-127 B

- *long definite length* (primitive form): the first byte (without leading 1) represents the length of the length field, that is, the number of octets necessary for encoding the length.

- *indefinite length* (constructed form): the first byte 1000 0000 indicates this form, the next octets represent the value of the length, and two zero octets (0x 00 00) are added after the encoding the value.

Example 2: sequence 61 81 83 80 1f 53... (hex)

- Type 61 (0110 0001 in binary) is an identifier octet which describes the application class (01), in the constructed (1) form with data type 1. Application type 01 means goosePDU.

- Length 81 83 is an extended length field where 0x81 (1000 0001) describes the long definite form of the length with 1 octet and 0x83 is the length value, that is, 131 bytes.

- Type 80 (1000 0000) starts an embedded TLV triplet which is of the context-specific class (10), primitive form (0) and the type is 0 which is gocbRef.

- Length 1f is the length of the VISIBLE STRING in the gocbReffield.

Encoding BIT STRING value
The encoding form of BIT STRING value can be primitive or constructed.

In the primitive form, the string is cut up in octets and a leading octet is added so that the number of bits left unused at the end could be identified by an integer between 0 and 7. If this octet is 0, it means that all bits are used.

For example, BIT STRING 1011 0111 0101 1 (13 bits) will be aligned to two octets (16 bits), e.g., 1011 0111 0101 1000, thus three zero bits are left added to align the bit string to octets. BER encoding will be *0000 0011* 1011 0111 0101 1000, where the first octet(italic)represents the number of left added zero bits (3) and two following octets represent the bit string without three last zero bits.

The *EXTERNAL* type

EXTERNAL data type is the first type that enabled the user to change the presentation context. It models values that are external to the current specification in the sense that they are defined with another abstract syntax or encoded with a transfer syntax different from that of the active presentation context. The component *direct-reference* identifies the data type syntax. The component *indirect-reference* is an integer that references one of the presentation contexts that were negotiated. The *data-value-descriptor* is a string that describes the abstract syntax of the data but it is not used in practice. For embedding the value in the *encoding* component, the item *single-ASN1-type* is chosen if the abstract syntax is an ASN.1 type and if the data are encoded with the same transfer syntax as the active presentation context.

```
EXTERNAL ::= [UNIVERSAL 8] IMPLICIT SEQUENCE {
    direct-reference        OBJECT IDENTIFIER       OPTIONAL,
    indirect-reference      INTEGER                 OPTIONAL,
    data-value-descriptor   ObjectDescriptor        OPTIONAL,
    encoding                CHOICE {
        single-ASN1-type        [0]     ANY,
        octet-aligned           [1]     IMPLICIT OCTET STRING,
        arbitrary               [2]     IMPLICITBIT STRING
    }
}
```

The type EXTERNALis used, for example, in the PDUs of the Association Control Service Element (ACSE) invoked by all the applications that use the OSI stack,