VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

DETEKCE ANOMÁLIÍ V SÍŤOVÉM PROVOZU POMOCÍ KOMPRESNÍCH METOD
DETECTION OF ANOMALIES IN NETWORK TRAFFIC USING COMPRESSION METHODS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Libor Blažek**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Blažek**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Libor Blažek ***ID:*** 136453
Ročník: 3 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Detekce anomálií v síťovém provozu pomocí kompresních metod

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je prozkoumat možnosti využití kompresních metod pro detekci anomálií v datové komunikaci a to zejména bezpečnostních útoků.
Úkolem je otestovat vhodnost jednotlivých metod komprese k tomuto účelu.

Dílčí cíle:
1. S použitím předepsané literatury a souvisejících informací na internetu zpracovat stručný stav řešení problematiky.
2. Vytvořit algoritmus vybraných kompresních metod a otestovat jejich funkčnost.
3. Porovnat zvolené metody a vybrat vhodnější pro analýzu datového provozu.
4. Zvolit a odsimulovat síťové útoky a otestovat je na zvolené kompresní metodě.

DOPORUČENÁ LITERATURA:

[1] SPEIDEL, Ulrich, Raimund EIMANN a Nevil BROWNLEE. Detecting network events via T-entropy. 2007 6th International Conference on Information, Communications. IEEE, 2007, : 1-5. DOI: 10.1109/ICICS.2007.4449642.

[2] SAYOOD, K. Introduction to Data Compression. 2nd ed. Morgan Kaufmann

Publishers, 2000. 636 s. ISBN 1-55860-558-4

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 1.6.2016

Vedoucí práce: Ing. Petr Blažek
Konzultant bakalářské práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

Abstrakt

Cílem bakalářské práce je návrh a praktická ukázka funkčnosti vybraných kompresních metod. V následujících kapitolách budou probrány útoky na koncová zařízení a zmíněna některá opatření. Na ukázku budou zpracovány dvě metody pomocí vývojového prostředí. Při útocích se bude zjišťovat anomálie v síti a následně se provede na jedné z metod ukázka komprese dat. Data budou zachytávaná v běžném provozu na koncové stanici a následně při útoku.

Summary

The objective of the thesis is to design a practical demonstration of the functionality of selected compression methods. The following chapters will discuss the attacks on terminals and mentioned some measures. The show will be processed using two methods development environment. The attacks will detect anomalies in the network and subsequently carried out at one of the sample data compression methods. Data will be collected as normal operation at the terminal station, and then in the attack.

Klíčová slova

DoS, DDoS, Wireshark, komprese, Huffman, LZ77, PING, hping3, flood, anomálie

Keywords

DoS, DDoS, Wireshark, compression, Huffman, LZ77, PING, hping3, flood, anomaly

BLAŽEK, L.*Detekce anomálií v síťovém provozu pomocí kompresních metod*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. ?? s. Vedoucí Ing. Petr Blažek.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma "DETEKCE ANOMÁLIÍ V SÍŤOVÉM PROVOZU POMOCÍ KOMPRESNÍCH METOD"jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení S 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně moných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno . . . .

(podpis autora)

Libor Blažek

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Petru Blažkovi, za odborné vedení, konzultace, trpělivost, podněty k bakalářské práci. Dále bych poděkoval panu Ing. Pavlu Snášelovi za konzultace k praktické části, zabývající se programováním kompresních metod.

Brno . . . .

(podpis autora)

Libor Blažek

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této bakalářské práci byl realizovaný v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systémů (SIX); registrační číslo CZ.1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

Obsah

1. Úvod

Cílem bakalářské práce je seznámit uživatele o možných útocích v oblastí informačních technologií. V první řadě je potřeba zjistit, jaké jsou možnosti a způsoby, jak může útočník prolomit zabezpečení, pokud by se chtěl dostat do vnitřních sítí, nebo jen znepřístupnit servery. Tyto možnosti jsou probrány v několika kapitolách. Jelikož předmětem není obrana proti těmto útokům, tak se v práci vyskytují ochrany proti těmto útokům jen informativně, které se nadále neprobírají.

Pomocí těchto útoků (bude zaměřeno na DoS a DDoS), vzniká v síti anomálie. Veškerá komunikace a také i případné anomálie lze zachytit pomocí některých programů. Z tohoto důvodu bude nainstalován a na praktickou část použit program Wireshark, který dokáže zachytávat komunikaci.

Zachycená data z komunikace a následně i útoku, jsou upravena tak, aby mohla být použita v další části práce, která se zabývá kompresní metodou.

V dnešní době existuje několik kompresních metod, které jsou využívány v jiných oblastech. Nejznámnější je například slovníková komprese, využívaná v balících programech RAR, nebo ZIP.

Vybrány jsou ale dvě metody a to LZ77 a Huffmanova komprese. Bude potřeba napsat program, který dokáže vstupní data převést na výstup pomocí těchto metod.

Na vytvoření těchto algoritmů bude využit program MS Visual studio 2015. Programovací jazyk je vybrán C#.

Jakmile budou programy napsány a základní funkčnost vyzkoušena na libovolném vstupním řetězci, je možné vybrat jednu kompresní metodu a vyzkoušet ji v případech běžné komunikace a následně při útoku DoS a porovnat tyto dvě výsledné komprese.

2. Možné útoky v síti

Kapitola bude věnována možným útokům. Budou probrány jen ty nejznámnější metody a dále se seznámíme s druhy, které existují podle útoku na jednotlivé vrstvy.

2.1. Kybernetické útoky

Jedná se o útok, který provádí skupina hackerů. Většinou je cílem těchto útoků ochromit cílový zdroj, získat data nebo dokonce vyřadit systém z provozu. Jsou prováděny pomocí tzv. zombie botů, což jsou nakažené počítačové sítě a koncová zařízení, kdy je téměř nemožné zjistit původ těchto útoků.[14]

Tyto akce podnikají lidé, kterým se říká „Hacker“, což jsou specialisté na počítačové sítě a programátoři. Dokážou využívat systémové chyby, které jsou v programech obsažené a upravit si je podle vlastních potřeb. Tato osoba se místo „Hacker“ správně označuje „Cracker“. První z termínů se používá masově v médiích, proto je také známější. Další pojmenování těchto osob jsou GEEK, GURU nebo NERD.[9]

Tito lidé ve většině případů napadají korporátní společnosti a vládní sektory, pokud se jim nelíbí o co tyto korporace usilují. Většinou jsou to služby proti volnému šíření informací, zpoplatnění některých služeb, vidina vlastního finančního prospěchu atd. Tyto osoby se rozdělují na několik druhů a také podle techniky, kterou využívají:[9]

White Hat - bílý kloubouk: Tato osoba se převážně zabývá zabezpečením operačního systému. Nepůsobí a nenapadá systémy za účelem poškodit, ale pouze zjistit chyby v systémech. Můžeme jej nazvat také jako „etický hacker“, který většinou spolupracuje na mezinárodní úrovni.

Black hat - černý kloubouk: Nyní se jedná o přesný opak White hat. Útočník se snaží prolomit zabezpečení za jediným účelem, a to škodit. Většinou ničí data nebo vyřadí počítačovou síť z provozu.

Gray hat - šedý kloubouk: Jedná se o spojení dvou předchozích druhů. Tato osoba může pronikat do systémů pouze za účelem informování správce sítě, že jejich systém byl napadený a za případný poplatek může nabídnout své služby k zabezpečení systému.

Elitní cracker: Jedná se o nejzkušenějšího člena, který se většinou musí prokazovat svými dovednostmi, pokud by chtěl být součástí širší skupiny. Tito členové většinou rozšiřují nově objevené exploity, což jsou programy, data, nebo příkazy, které dovolují získat výhodu nad daným programem nebo systémem.

2.2. Techniky crackerů

2.2.1. Skenování portů

Jedná se o princip, kdy útočník zjišťuje, které porty jsou průchozí na vzdálenem počítači. K tomu pomahají speciální programy jako je NMAP (Network MAPer). Jedná se o bezpečnostní sken portů. Program byl vytvořen na skenování velkých počítačových sítí,

kde dokáže zjistit přítomnost a pravidla firewallu, jaké porty jsou využívány pro služby a další informace. Je několik druhů skenování:

TCP: Skenují se porty metodou dotaz a odpověď. Volá se příkazem „connect()“ a pokud dojde pozitivní odpověď, tak je port otevřen a může být sledován. V opačném případě přijde chyba. Tento postup může provádět kdokoliv, je však velmi rychle vypátratelná.

SYN: Jedná se o TCP skenování, kdy je posílán paket SYN a pokud je port otevřen, dojde odpověď SYN-ACK. Jakmile dojde tato zpráva, skaner pošle RST, což je žádost o ukončení spojení, a poté se komunikace na port ukončí. Tím pádem není tento útok tak lehce zjistitelný.

UDP: Jedná se o velice pomalou a ne moc spolehlivou metodu. Pokud jsou porty blokovány firewallem, budou přicházet nekorektní odpovědi.

ACK: Díky této metodě se dá přesně zjistit, zda jsou porty otevřené nebo zavřené a také, jestli je filtrován přes firewall

2.2.2. Prolamování hesel

Tento způsob získávání přístupu může být z hlediska složitosti hesel velmi zdlouhavý. Útočník, který používá tuto metodu, musí spoléhat na uživatele, že nepoužil příliš složité heslo, aby jej mohl získat. Využívá se nejčastější metoda „Brute Force“, kdy jsou hesla zkoušena podle vytvořených „knihoven hesel“, která obsahuje nejznámější používaná hesla a poté zkouší veškeré kombinace.

2.2.3. Zachytávání paketů

Pokud útočník dokáže scanovat pakety v síti a zachytávat je, vzniká velká pravděpodobnost, že může odchytit citlivé údaje (login a heslo). K zachytávání paketů slouží spousta programů, například „Wireshark“, který dokáže získat informace o komunikaci po síti.

2.2.4. Phishing

Velmi rozšířený způsob, jak získat od napadených osob přístup, bez použití složitějších programů a napadání systému. Vytváří se většinou co nejpřesnější repliky webových stránek, nebo programů, které jsou považovány za důvěryhodné. Nejčastěji bankovní přístupy nebo platební brány, kde klient sám a dobrovolně, aniž by si zkontroloval pravost stránek (lze poznat z „URL“) zadá veškeré přihlašovací údaje, nebo informace o platební kartě.

V dalších případech jsou používány počítačové viry, které se dokáží šířit bez vědomí uživatele. Z pravidla bývá umístěn v souborech, jako jsou stažené programy, hry, nebo filmy. Útočník může používat také program, například „Keylogger“, který zaznamenává stisknuté klávesy a tyto informace se posílají útočníkovi.

Jeden z nejznámnějších druhů útoků na servery, nebo konkrétní aplikace je DoS nebo DDoS útok.

3. DoS vs. DDoS

V prvním případě je potřeba rozlišit útoky typu DoS (Denial of Service) od DDoS (Distributed Denial of Services). V obou dvou případech se jedná o útok se stejným cílem, liší se pouze zdrojem, odkud se požadavky na server generují. Jak už je z názvu „Distributed“ patrné, jedná se o způsob posílání z různých zdrojů, a proto je detekce takových útoků velmi náročná protože nelze zjistit zdroj. Oproti DDoS využívá útok DoS pouze jeden zdroj, proto při lepším zabezpečení lze využít způsob obrany blokací určité IP adresy, která zahlcuje systém.

Sousloví DDoS je složeno ze dvou částí, kdy jedna část je ze slova "Distributed", což znamená „distribuovaný“ – rozložený. Vychází z nutností použití více prvků, jelikož je tak jednodušší generovat více dotazů, proto je také hůře dohledatelný.[17]

DoS znamená v překladu odmítnutí služby, kdy je hlavním cílem těchto útoků vyřadit konkrétní servery, aby nebylo možné poskytovat služby. Server bude v tomto případě nedostupný.[15]

V praxi funguje DDoS tak, že skupina uživatelů se domluví a v jednu chvíli si budou chtít prohlédnout konkrétní stránku. Pokud jich bude dostatečný počet, tak aby zátěž, která je vyvolávaná byla větší než zvládne server, nestihnou se požadavky zpracovávat a stránku nepůjde zobrazit.

Jestliže na serveru běží i jiné služby, popřípadě aplikace, je možné, že v případě takového útoku nebudou dostupné, zvláště pokud je zabezpečení špatně nakonfigurováno. Pokud není server nakonfigurován tak, aby po těchto útocích sám restartoval služby, nemusí být funkční ani po ukončení těchto útoků. Potom je potřeba, aby se o zprovoznění postaral správce.

Musíme si uvědomit, že ať název „DDoS útok“ zní negativně, tak se nejedná o útok s cílem něco odcizit, nebo nějak využít pro svůj prospěch. Tento tzv. „útok“ slouží pouze k zahlcení serveru tak, aby nemohl nadále poskytovat služby. V tomto případě se nemusí vždy jednat o trestný čin, pokud nebude v daném státě v rozporu se zákonem a nezpůsobí se tím prokazatelné škody.

3.1. Útoky na jednotlivé vrstvy

Pokud si útočník vybere DoS nebo DDoS metodu, tak jsou tyto útoky ještě členěny na další typy útoků, podle OSI vrstvy, na kterou se bude útok provádět. Záleží podle toho, co by útočník chtěl vyřadit z provozu. Jelikož OSI vrstvy jsou tvořeny ze 7 částí, tak zabezpečení jedné vrstvy nezabrání útoku na jakokoliv jinou vrstvu, proto je nutné každou z těchto 7 vrstev zapezpečit zvlášť.[15] [16]

Příklady některých útoků na jednotlivé vrstvy

Aplikačni vrstva

Útoky: FTP, DNS, DHCP, SMTP, SSH, TELNET

Zmírnění útoku: Jelikož se jedná o aplikační vrstvu, na které pracuje uživatel, je těžké odhalit, jestli se jedná o útok (napodobuje lidské chování), proto je ochrana náročnější.

Prezentační vrstva

Útoky: Komprimace, šifrování, konvertování Útočí se pomocí SSL dotazů, které jsou upraveny.

Zmírnění útoku: Ochrana možná přesměrováním SSL dotazů přes jiný zdroj (z původní infrastruktury).

Relační vrstva

Útoky: Využívá se prtokol na zahájení a ukončení spojení. Omezují se služby běžících jinak než přes Telnet.

Zmírnění útoku: Pravidelně provádět update SW.

Transportní vrstva

Útoky: TCP a UDP protokoly. Omezení síťových připojení.

Zmírnění útoku: Je potřeba informovat poskytovatele připojení o napadení.

Síťová vrstva

Útoky: Směrovací protokoly, ICMP flooding.

Zmírnění útoku: Je potřeba stanovit limity pro ICMP.

Linková vrstva

Útoky: Aktivní prvky (switch/router), MAC flooding.

Zmírnění útoku: Omezení MAC adres, které jsou přípustné, vytvoření pravidel pro přístup.

Fyzická vrstva

Útoky: Síťové kabely, fyzické manipulace s vodičem.

Zmírnění útoku: Zamezit volný přístup k infrastruktuře.

4. Monitorování provozu

Pokud správce sítě (administrátor) nemonitoruje počítačovou síť, nemůže nikdy zjistit, jaké útoky probíhají v infrastruktuře. Je také dobré sledovat, co ze sítě odchází/přichází, aby bylo možné následně vytvořit zabezpečení. Je také potřeba tyto data uchovávat a provádět diagnostiku provozu sítě i v řádech několika měsíců zpět.

K tomuto sledování je vytvořeno několik programů, které dokáží sledovat tok informací v síti, vytížení procesů a dokáží zachytávat pakety (ukládat a dále je zkoumat). Jsou schopné provést hloubkové analýzy serverů/stanic. Mohou sledovat činnosti routeru, firewallu.

Mezi nejznámnější programy pro tuto práci, kterou využívají jak začínající správci, amaterští uživatelé nebo také profesionálové ve firemních sítích, je Wireshark. Jsou zde také ostatní programy jako OpManager nebo Network Monitor.[18]

4.1. Wireshark

Jedná se o program (známý také pod názvem Ethereal), který slouží k analýze protokolů a paketů. Nejčastěji se využívá v počítačových sítích, za účelem vývoje nebo studiem komunikace mezi prvky. Program je vytvořen pro téměř všechny operační systémy - Linux, Mac OS X, Windows. Wireshark má grafické rozhraní kde je možnost filtrovat odchozí nebo příchozí komunikaci, dokáže přepínat síťové karty do promiskuitního režimu (dokáže předávat systému veškerá data která obdrží). Program je také v terminálové verzi „Tshark“ pro Linux a veškerý software je vydáván jako „Open Source“.[19]

Díky tomu, že program dokáže zpracovávat různé síťové protokoly, je schopen zobrazit veškeré zapouzdření pro různé vrstvy. Používá se knihovna „pcap“ k zachycení komunikace a paketů, které tato knihovna podporuje.

4.2. Network Monitor

Slouží ke sledování v nepřetržitém provozu lokální sítě, koncových stanic a služeb. Aplikace dokáže upozornit na odchylky a vygeneruje příšlušný report o selhání. Mohou se nastavit objekty, které sledují danné prvky nebo služby. Uživatel vytváří pravidla, která mají být dodržována a v případě nedodržení budou hlásit chybu, ta může být zaslána prostřednictvím emailu, SMS, nebo také restartuje stanici.

4.3. Ochrana proti (D)Dos útokům

V dnešní době jsou prováděny útoky hlavně na velké společnosti jako je Microsoft se svým herním systémem, SONY a filmové společnosti. Cílem útoků je znemožnit přístup na servery a využívat služeb (online hraní, sledování filmů). Proto je hlavním cílem těchto, ale i dalších firem, zabezpečit systémy před takovým nebo podobným útokem. Jedna z možných ochran proti DoS útokům na UDP protokoly analyzovala hostingová firma OVH. Zaměřila se na herní průmysl a také komunikační aplikace. Byl vytvořen filtr, který analyzuje příchozí a odchozí přenos dat. Jako hlavní předností systému je schopnost vyfiltrovat dotazy připojených uživatelů od ostatních, které jsou součástí útoku. Chová se tak, že data zpracovává v paměti cache a poté filtruje TCP/IP a UDP pakety.[18]

5. Anomálie

Abychom správně detekovali anomálie, je potřeba také zajistit dostatečný sběr dat v síťovém provozu. O tyto sběry se mohou postarat sondy NetFlow, přepínače nebo směrovače. Díky dalším programům je možné tyto data zobrazovat v grafech či tabulkách. Žádný člověk není schopen diagnostikovat tak velké množství dat, aby detekoval vzniklé anomálie, proto jsou vytvořené nástroje, díky nimž je detekce anomálií jednodušší.

Jakmile jsou nalezeny výkyvy v datech, je potřeba se na tento problém zaměřit a provést vyhodnocení, které umožní rychlou nápravu. Pokud by se projevila v síti porucha zařízení (router), může dojít k již zmíněné anomálii, která může ovlivnit infrastrukturu. Projevit se může špatnou odezvou, zpožděním, tím že prvky nebudou mít aktuální informace v DNS tabulce. Na druhou stranu, může být problém také způsoben viry nebo spamem, který dokáže nadělat velké škody, pokud není zavčas detekován a opraven. V tomto případě je potřeba dodržovat veškerý software aktuální a dodržovat zásady bezpečnosti.

5.1. Provozní anomálie

V počítačové síti probíhá standartně komunikace mezi prvky, pokud není detekován žádný problém. Pokud by se v průběhu těchto komunikací poškodil nějaký aktivní prvek, nebo se v důsledku zahlcení sítě zpomalil provoz, tak můžeme tvrdit, že se jedná o provozní anomálii.

5.2. Bezpečnostní anomálie

Již podle názvu, lze usoudit že se bude jednat o bezpečnostní provoz v danné síti. Tyto anomálie mohou být způsobeny nakaženou stanicí virem nebo jiným škodlivým softwarem. Posléze můžou způsobit datové úniky a trhliny v bezpečnostních systémech. Proto je potřeba udržovat software aktuální.[13]

5.3. Způsoby detekce anomálií

5.3.1. Porovnávání signatur

Metoda označována také jako NIDS (Network Intrusion Detection Systém). Jedná se o detekci průniku. Tyto signatury jsou pozorovány během známého útoku a v případě shody, může NIDS upravit pravidla na firewallu nebo QoS. Tato metoda porovnává síťový provoz na úrovni paketů.

5.3.2. Stavová analýza

Je potřeba přesně definovat protokoly provozovaných na síti, které jsou v dokumentech RFC či jiných standartech. Pokud se nebude provoz na síti chovat podle předem definovaných protokolů, bude brána komunikace v síti za anomálii.

5.3.3. Behaviorální analýza

Jedná se o vzor, který je vytvořen a následně se porovnává s provozem v síti. Pokud provoz nesouhlasí s takovým pravidlem, je brána jako anomálie. Je nevýhodou, pokud jsou pravidla upravována, že se může projevit anomálie.

6. Kompresní algoritmy

Hlavním cílem kompresních algoritmů (metod) je co nejvíce zredukovat objem dat, ale přitom zachovat jejich informační hodnotu. Někdy však dojde ke ztrátám, které při dekompresi nejsou kritické, ale je lepší komprimovat bezztrátově. Slouží hlavně k přenosu dat nebo archivaci.

Od komprese se také odvíjí kvalita a rychlost. Například při symetrickém algoritmu je čas potřebný na kompresi a dekompresi shodný oproti asymetrickému algoritmu.

Nejdůležitější vlastností je právě ztrátovost/bezztrátovost komprese a také kompresní poměr, který je dán poměrem výstupního k vstupnímu řetězci. Aby byla komprese „úspěšná“, musí být výsledná hodnota menší než jedna. Pokud by byla rovna jedné, tak nedochází k žádné kompresi, případně expanzi, pokud je větší než jedna.[12]

$$K < 1, K = 1, K > 1$$

Další z pojmů je analogická veličina, kde označujeme kompresní zisk, který udává hodnotu ušetřeného prostoru (dat):

$$Z = 1 - K$$

Bezeztrátová komprese po dekódování, má shodný výsledek s původním stavem. Používá se převážně při kompresi textů.

Ztrátová komprese má výsledek po dekódování, který nesouhlasí se zdrojem. V této kompresi proběhla ztráta. Většinou se využívá při kompresi obrazů nebo zvuků.

6.1. Entropie

Entropie udává množství informací v danné zprávě. Jako první se touto problematikou zabýval Claude Shannon, který jako první k této veličině přiřadil jednotku a tou je bit. Zjednoduššeně by se mohlo říct, že entropie jsou otázky, na které se dá odpovědět pouze ANO/NE, a tím by jsme mohli odhalit obsah zprávy. Pokud bychom měli konečnou množinu S (zdrojová zpráva) a pravděpodobnostní distribuci P(s), je entropie Hi definována záporným logaritmem její pravděpodobnosti.

$$S_i = \{S_1, S_2, S_n\} P_i = \{P_1, P_2, P_n\} H_i = -\log\{2\} P_i$$

Entropie kódu [**K**] je vážený průměr entropie zdrojových jednotek vážený jejich pravděpodobností:

$$H_k = \sum S \in S H_s * P_s H_k = \sum s \in S H_s * P_s$$

6.2. Slovníkové metody

Základní princip spočívá ve vyhledávání shodných a opakujících se řetězců v daném dokumentu. Tyto řetězce jsou pak na výstupu již komprimovány a jsou jim přiřazeny kódy. Při průchodu daným souborem se slovník vytváří samostatně, ale tak, aby bylo možné data opět převést zpět do původní formy. Na výběr je několik metod, které jsou vytvořeny, ale je důležité vybrat také tu správnou, která má správné vlastnosti. Mohou být metody ztrátové(JPEG) nebo neztrátové LZW nebo LZ77. Dále se dělí na asymetrické, nesymetrické, jedno nebo více průchodové.[3]

7. LZ77

Kompresní algoritmus vytvotřili Abraham Lempel a Jacob Ziv roku 1977. Metoda je slovníkového typu a používá tzv. „klouzavé okno“. Okno se rozdělí na dvě části, kde jedna bude již prohlížecí a druhá bude sloužit jako aktuální. Velikost těchto podoken je konstantní a malá. Délka aktuálního okna musí být vždy menší nebo rovna oknu prohlížejícímu. Hlavní výhodou je to, že náročnost nesouvisí s velikostí vstupního řetězce, pracuje se pouze s daty umístěnými v oknech.[12]

7.1. Komprese

V první řade se z pole odstraní veškeré mezery a vytvoří se jednotný řetězec, který se následně přiřadí do části aktuálního okna a prohlížecí okno bude zatím prázdné (se zbytkem řetězce se bude pracovat postupně). V každém kroku se bude vyhledávat shodné nejdelší slovo, které je v prohlížecím okně a zároveň v okně aktuálním (zde musí být vždy začátkem první znak - prefix). Jakmile je nalezena shoda, nalezené slovo bude zakódováno (i,j,z).[1]

i: Jedná se o hodnotu, kterou má znak v prohlížecí tabulce, tato hodnota je počítána od konce tabulky.

j: Délka slova, nebo-li počet znaků, která jsou shodná s aktuální tabulkou.

z: Znak, který následuje za slovem.

Jakmile je slovo zakódováno, posune se celý řetezec o hodnotu j+1 doprava. Tento postup se opakuje dokud není v aktuální tabulce prázdný řetězec.

7.1.1. Příklad komperse

Příklad komprese bude ze slova ABRAKADABRA.[7] V prvním kroku se načte řetezec do aktuálního okna, které bude obsahovat 3 znaky. Porovnávat se bude s prohlížecím oknem, které je momentálně prázdné a bude obsahovat znaky 4. Dále bude zaznamenán posun a výstup.[10]

Tabulka 7.1: Komprese pomocí LZ77

Krok 1	**Posun**	**Prohlížecí okno**	**Aktuální řetězec**	**Výstup**	**Zbytek řetězce**
Načtení řetezce	0	- - - -	ABR	-	AKADABRA

Krok 1

Podle prvního kroku 7.1 se načetly znaky „ABR“ a jelikož nebylo s čím porovnat, tak výstupní hodnota je prázdná (zaznamenává se z předchozího kroku), posun také nebyl proveden, zbytek řetezce je umístěn na konci tabulky.

Tabulka 7.2: Komprese pomocí LZ77

Krok 2	**Posun**	**Prohlížecí okno**	**Aktuální řetězec**	**Výstup**	**Zbytek řetězce**
Načtení řetezce	1	- - - A	BRA	-	KADABRA

Krok 2

Hodnota „j“ byla nulová, tudíž se provedl posun o jednen znak. Znak, který následuje je „A“, zobrazený momentálně v prohlížecím okně. 7.2

Tabulka 7.3: Komprese pomocí LZ77

Krok 3	**Posun**	**Prohlížecí okno**	**Aktuální řetězec**	**Výstup**	**Zbytek řetězce**
Načtení řetezce	1	- - A B	RAK	0,0,B	ADABRA

Krok 3 a 4

Z kroku 2 opět porovnáváme aktuální řetezec „BRA“ s hodnotou v prohlížecím okně „- - - A“. Jelikož prefix aktuálního řetězce neodpovídá žádnému znaku v prohlížecím okně, tak bude opět posunut řetězec o jeden znak vlevo. 7.3 7.4

Tabulka 7.4: Komprese pomocí LZ77

Krok 4	**Posun**	**Prohlížecí okno**	**Aktuální řetězec**	**Výstup**	**Zbytek řetězce**
Načtení řetezce	1	- A B R	AKA	0,0,R	DABRA

Tabulka 7.5: Komprese pomocí LZ77

Krok 5	**Posun**	**Prohlížecí okno**	**Aktuální řetězec**	**Výstup**	**Zbytek řetězce**
Načtení řetezce	2	B R A K	ADA	3,1,K	BRA

Krok 5

V předchozím kroku 7.4, si můžeme všimnout, že prohlížecí okno je tvořeno znaky „ABR“ a aktuální řetězec „AKA“. Prefix tohoto okna je shodný s prvním znakem prohlížecího. Shoduje se pouze jeden znak, tudíž hodnota „j“ bude rovna 1. Můžeme si všimnout že hodnota „i“ je 3, ikdyž je shodný znak „A“ jako první v prohlížecím okně a to protože se pozice znaku počítá zprava. Řetězec se posunul o dva znaky. 7.5

Tabulka 7.6: Komprese pomocí LZ77

Krok 6	**Posun**	**Prohlížecí okno**	**Aktuální řetězec**	**Výstup**	**Zbytek řetězce**
Načtení řetezce	2	A K A D	ABR	2,1,D	A

Krok 6

Z kroku viz tabulk 7.5 je vidět opět shodný prefix aktuálního řetězce s prohlížecím oknem. Na výstupu bude tedy hodnota i=2, jelikož se jedná o znak „A“, který je na druhé pozici, shodný znak je pouze jeden a následuje znak „D“. Opět řetězec posuneme o dva znaky. 7.6

Tabulka 7.7: Komprese pomocí LZ77

Krok 7	Posun	Prohlížecí okno	Aktuální řetězec	Výstup	Zbytek řetězce
Načtení řetezce	2	A D A B	RA-	4,1,B	-

Krok 7

Jak v kroku 6 7.6 i zde shodný řetězec pouze znak „A“, bude tudíž hodnota j=1, následující znak bude „B“ a bude shodný s prvním znakem v prohlížecím okně a proto je hodnota i=4. 7.7

Tabulka 7.8: Komprese pomocí LZ77

Krok 8	Posun	Prohlížecí okno	Aktuální řetězec	Výstup	Zbytek řetězce
Načtení řetezce	1	D A B R	A - -	0,0,R	-

Krok 8

Jedná se o stejný případ jak v prvních krocích, kdy není shodný žádný znak. Proto bude posun o jeden znak a následující znak bude „R“. 7.8

Tabulka 7.9: Komprese pomocí LZ77

Krok 9	Posun	Prohlížecí okno	Aktuální řetězec	Výstup	Zbytek řetězce
Načtení řetezce	2	A B R A	- - -	3,1,-	-

Krok 9 V posledním kroku vidíme v aktuálním řetězci znak „A“, ten je shodný s prvním znakem v prohlížecím okně, proto bude mít výstup hodnotu i=3, j=1 a již nenásleduje žádný další znak proto z=0. 7.9 Výsledný řetězec bude v tomto případě vypadat následovně:
Výstup=(0, 0, A) (0, 0, B) (0, 0, R) (3, 1, K) (2, 1, D) (4, 1, B) (0, 0, R) (3, 1, konec)

7.1.2. Následná dekomprese

Vychází se z výstupu zakódovaného slova a začíná se podle prvního kroku při kódování řetězce 7.1. Tabulka bude obsahovat jiné kroky.

Tabulka 7.10: Deomprese pomocí LZ77

Krok 1	Vstup	Řetězec	Načteno	Výstup
Vstupní kód	0,0,A		A	A

Na začátku je prázdný řetězec a podle hodnoty vstupu přidáme znak „A“ na první pozici. 7.10

Tabulka 7.11: Deomprese pomocí LZ77

Krok 2	Vstup	Řetězec	Načteno	Výstup
Vstupní kód	0,0,B	A	B	AB

Z předchozího kroku je již načten řetězec a k němu se přidá další znak, stejně i v následujícím kroku. 7.11 7.12

Tabulka 7.12: Deomprese pomocí LZ77

Krok 3	**Vstup**	**Řetězec**	**Načteno**	**Výstup**
Vstupní kód	0,0,R	AB	R	ABR

Tabulka 7.13: Deomprese pomocí LZ77

Krok 4	**Vstup**	**Řetězec**	**Načteno**	**Výstup**
Vstupní kód	3,1,K	ABR	AK	ABRAK

Zde je v paměti již řetezec „ABR“ podle vstupu následuje znak, který je umístěn na třetí pozici, což je momentálně znak ”A”a za ním bude znak „K“. 7.13

Tabulka 7.14: Deomprese pomocí LZ77

Krok 5	**Vstup**	**Řetězec**	**Načteno**	**Výstup**
Vstupní kód	2,1,D	ABRAK	AD	ABRAKAD

Zde 7.14, jako v předchozím případě je k řetězci přiřazen znak podle vstupu „A“ a k němu první následující „D“. Stejným způsobem se pokračuje do té doby, dokud je na vstupu v proměnné hodnota znaku. Pokud není žádný znak jedná se o konec řetezce viz následující kroky. 7.15 7.16 7.17

Tabulka 7.15: Deomprese pomocí LZ77

Krok 6	**Vstup**	**Řetězec**	**Načteno**	**Výstup**
Vstupní kód	4,1,B	ABRAKAD	AB	ABRAKADAB

Tabulka 7.16: Deomprese pomocí LZ77

Krok 7	**Vstup**	**Řetězec**	**Načteno**	**Výstup**
Vstupní kód	0,0,R	ABRAKADAB	R	ABRAKADABR

Tabulka 7.17: Deomprese pomocí LZ77

Krok 8	**Vstup**	**Řetězec**	**Načteno**	**Výstup**
Vstupní kód	3,1,-	ABRAKADABR	A	ABRAKADABRA

V posledním kroku přidáme konečné znaky a podle vstupu vidíme že již není žádný znak, který by následoval, proto bude ukončeno dekódování a bude zobrazen celý výstupní řetězec.

8. LZW

Nejrozšířenější metodou je LZW, kterou vyvinuli v roce 1984 Lempel, Ziv a Welch, proto také zkratka LZW. Nyní slouží v základních programech pro kompesi, jako je třeba ZIP, v grafice to jsou formáty GIF a TIFF nebo také PDF. Vychází z metody, kterou již vyvinuli dříve (LZ78), která částečně vychází z metody LZ77. Metoda je jednoduchá na implementaci, ale je o to náročnější na paměťovou velikost slovníku. K tomu je možné využít pravidla, která zamezí nárust slovníku nad definovanou velikost tím, že se budou odmazávat fráze, které již nejsou používané. Metoda je jednoprůchodová, slovníková a symetrická.[11]

8.1. Komprese

Jelikož se jedná o slovníkovou metodu, je nutné mít vytvořený slovník, se kterým se bude řetězec porovnávat. Z pravidla bývá tento slovník tvořen všemi dostupnými znaky (počet znaků z rozsahu 0-255). Pokud by byla následující hodnota 256, tak by představovala nový slovník a 257 konec slovníku. Pro ukázku bude vytvořena abeceda jen o pár znacích (A, B, C, D), aby příklad nemusel obsahovat rozsáhlé tabulky.[8] Je potřeba seřadit znaky abecedně a přidat jim pořadové číslo (kód):

A=0, B=1, C=2, D=3

Dále bude nutné mít vstupní data. Jelikož naše abeceda obsahuje pouze 4 znaky, bude tvořen následující řetězec:

Vstup = abacdacacadaad

Tabulka 8.1: Komprese pomocí slovníkové metody.

Krok	Vstup	Starý znak	Kód	Nová fráze	Nový kód
1	abacdacacadaad	a	0	ab	4

Můžeme z tabulky 8.1 zaznamenat vstupní řetězec, který na začátku obsahuje znak „a“. Ten je v abecedě na první (nulté) pozici. Nově se vždy vytvoří do slovníku nová fráze, která obsahuje znak z původního slova, která se porovnávala a přidá se další znak, který následuje, viz tabulka a následně se přidá index(nový kód), který pokračuje. Nesmí se také zapomenout odmazat znak, nebo znaky, podle nalezené fráze. [8] [20] V prvních pěti krocích 8.2 je nalezen vždy první znak, proto se vytvoří nové slovníkové fráze o dvou znacích a vždy se odmaže první jeden znak, který byl nalezen.

Tabulka 8.2: Komprese první části vstupu.

Krok	Vstup	Starý znak	Kód	Nová fráze	Nový kód
1	abacdacacadaad	a	0	ab	4
2	bacdacacadaad	b	1	ba	5
3	acdacacadaad	a	0	ac	6
4	cdacacadaad	c	2	cd	7
5	dacacadaad	d	3	da	8

V dalším kroku viz tabulka 8.3, je vstupní řetězec tvořen znaky: acacadaad. Podle již vytvořeného slovníku, je shoda na indexu 6 (ac). Následující znak za tímto řetězcem je „a“, proto do slovníku bude zaznamenána nová fráze „aca“. Nyní se odmažou první dva znaky, jelikož byly shodné se slovníkovou frází.

Tabulka 8.3: Komprese zbytku vstupního řetězce.

Krok	Vstup	Starý znak	Kód	Nová fráze	Nový kód
6	cacadaad	ac	6	cd	9
7	acadaad	aca	9	acad	10
8	daad	da	8	daa	11
9	ad	a	0	ad	12
10	d	d	3	—	—

Nyní porovnáváme vstupní text „acadaad“. Ten obsahuje slovo z indexu 9 (aca). Následuje znak „d“, který se přidá k nalezené frázi a vytvoří se nové slovo. Vymažou se v tomto případě 3 znaky.

Tímto způsobem se pokračuje do konce tabulky, kde bude výstup ve tvaru:

Výstup = 0102369803

8.2. Dekomprese

Z kompresní metody byl získán výstupní řetězec a slovník, který byl touto metodou vytvořen. Díky těmto údajům je možné dekódovat údaje a znovu vytvořit vstupní řetězec, použitý na kompresi.

Nyní je potřeba načíst vstupy, v pořadí, v jakém se komprimovala. První 4 znaky (původní slovník), které jsou shodné i při kompresi budou mít index opět 0 – 3. V prvním kroku bude na vstupu hodnota „0“ (výstup při kompresi)8.4. Hodnota „0“ představuje znak „a“. Neobsahuje žádný index, jelikož není vytvořená žádná fráze.

V druhém kroku je hodnota „1“, tvořena znakem „b“. Vytvoří se nová fráze způsobem, že se vezme fráze z minulého kroku a přidá se první znak z fráze aktuálního kroku. Proto se vytvořila nová fráze „ab“.

Dále se pokračuje až do konce stejným způsobem. Ukážeme si ještě jeden z kroků, kdy je na vstupu hodnota 9. Z předešlé tabulky 8.3 lze zjistit, že na výstupu byl řetězec „ac“, ten se přenese i do nové fráze a víme, že hodnota 9 je tvořena znaky „aca“. Vezme se první znak a přidá se k nové frázi.

Na dalším kroku je vstup 8 tvořen znaky „da“. Z předchozího kroku bude v nové frázi „aca“ + „d“.

Tabulka 8.4: Dekomprese pomocí slovníkové metody.

Krok	**Vstup**	**Výstup**	**Nová fráze**	**číslo**
1	0	a	—	—
2	1	b	ab	4
3	0	a	ba	5
4	2	c	ac	6
5	3	d	cd	7
6	6	ac	da	8
7	9	aca	aca	9
8	8	da	acad	10
9	0	a	daa	11
10	d	d	ad	12

Tímto způsobem se provede dekomprese řetězce. Můžeme podle výstupu zjistit, že se hodnota výstupního řetězce rovná vstupnímu řetězci při prováděné kompresi:

Výstup = abacdacacadaad

9. Huffmanovo statické kodování

Huffmanovo kódování je jedno ze základních a také rozšířených metod komprese dat. Z této metody vzniklo několik známých programů, které jsou používány do dnešní doby. Obdoba Huffmanovy metody je také Shannon-Fanův kód, který vytváří kódy shora dolů a od levého nejvyššího bitu a Huffmanův kód se generuje zdola nahoru a počítá se od nejnižší četnosti. Huffman na tento model přišel v roce 1952, kdy byla jedna z hlavních otázek výzkumu komprese dat.[4] [2]

Jako již vystudovaný magistr Huffman publikoval práci, ve které popisoval tuto metodu (Metoda pro vytvoření kódu s minimální redundancí). Tuto práci a výzkum však nikdy nepatentoval.

V prvním kroku je potřeba vytvořit seznam znaků (symbolů), které tvoří danný řetězec. Tento řetězec je potřeba seřadit dle četnosti výskytu a to od nejnižší po nejvyšší. Následně se bude vytvářet „strom“, kdy se budou spojovat vždy dva znaky s nejnižší četností a ty se v novém „listu“ sečtou a zaznamenají se směrem nahoru. Tímto postupem zredukujeme celý řetezec znaků až k vrcholu „stromu“ kdy zbyde pouze jeden znak s nevyšší četností, která bude zastupovat veškeré znaky.

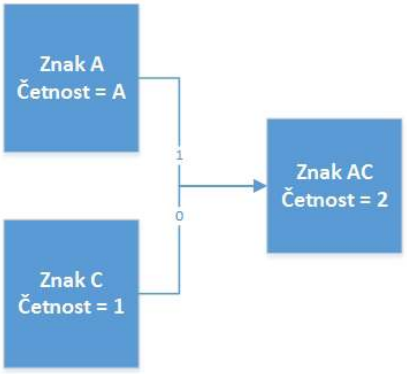
9.1. Příklad

K vytvoření stromu bude potřeba řetězec znaků a hodnoty, které odpovídají celkovému počtu v řetězci. Dejme tomu, že bude tvořen znaky “A“, “C“,“E“ ,“N“,“T“,“U“,“X“ a dále je nutné přiřadit pravděpodobnost (četnost) a seřadit je od nejnižší.[5] [6]

A = 1, C = 1, T = 2, N = 2, X = 5, E = 6, U = 7.

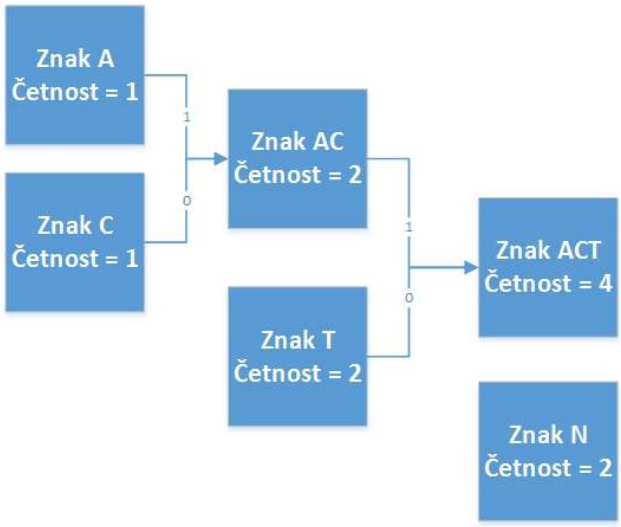
V Huffmanově metodě se vytváří strom od spodu nahoru(nebo libovolně, jak je zrovna potřeba), proto se vypíšou znaky s četností vedle sebe (pod sebe). Vezmou se dva znaky s nejnižší četností a vytvoří nový „list“, který je bude reprezentovat s hodnotou, představující jejich počet po sečtení. 9.1

Spojnice, jenž spojuje nový list s původními, budou číslovány hodnotou “0“ nebo “1“. Veškeré spojnice, směřující nahoru doprava budou mít hodnotu “1“ a ostatní “0“, nebo opačně, ale je nutné pravidlo dodržet po celou dobu.



Obrázek 9.1: Huffmanovo kódování - spojení dvou znaku s nejnišší četností.

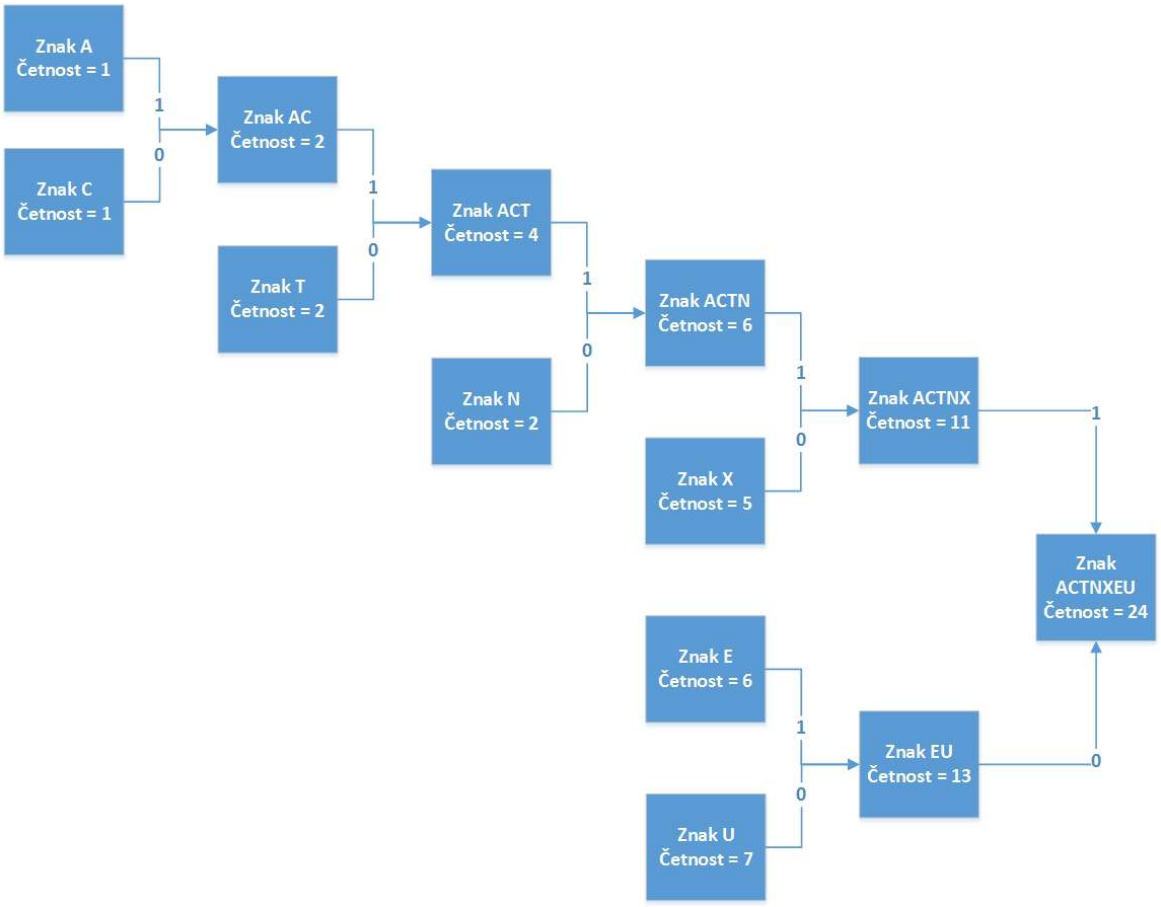
Znaky s nejnižší pravděpodobnosti v aktuálním „stromu“ jsou ”A”a ”C”. Po následném spojení bude mít vytvořený „list“ označení „AC“ a pravděpodobnosti jsou sečtené, tudíž hodnota je rovna 2. 9.1 Nyní jsou tři znaky tvořeny stejnou četností (“AC“,“T“ a “N“). V dalším kroku můžeme spojit libovolné znaky s nejnižší pravděpodobností, pro příklad budou vybrány znaky “AC“ a “T“, kdy po sečtení vznikne nový znak “ACT“ s hodnoutou četnosti 4. 9.2



Obrázek 9.2: Huffmanovo kódování - výběr jednoho znaku a následné spojení.

Jak může být již vidět podle obr. 9.2, pokud by bylo vybráno k hodnotě “AC“ druhý znak, tak by jsme došli ke stejnému závěru. Tímto způsobem se vytvoří „strom“ až k poslednímu znaku 9.3, reprezentující celou abecedu tvořenou řetězcem. Nesmí se zapomínat také na označení spojnic hodnotami “0“ a “1“, aby mohl vzniknout kód. Ikdyž byl „strom“ tvořen zespodu, tak kód, který bude vytvářen, musí být počítán od vrcholu, tudíž posledního znaku, k danému znaku který bude zakódován.

Při vytváření těchto „stromů“ se stejnými znaky a četností, mohou při rozhodování vzniknout odlišné výsledné kódy, ale střední délka kódu vždy zůstane. Odlišnost může vzniknout v případě kroku, kdy bylo ke znaku “AC“ přidán znak “N“ namísto hodnoty “T“. 9.2



Obrázek 9.3: Huffmanovo kódování - kompletní Huffmanův strom.

Pro porovnání můžeme vytvořit dvě tabulky 9.1 a 9.2, kdy v jednom případě bude použito klasické nahrazení znaku na 4bitový kód a v druhé tabulce Huffmanovo kódování.

Tabulka 9.1: Tabulka četností znaků a přiřazení binární hodnoty.

Znak	**Četnost**	**Kód**
A	1	0000
C	1	0001
T	2	0010
N	2	0011
X	5	0100
E	6	0101
U	7	0110
Velikost kodu(bit)	suma=24	81

Tabulka 9.2: Tabulka četností znaků a přiřazení binární hodnoty - Huffmanovo kódování.

Znak	**Četnost**	**Kód**
A	1	11111
C	1	11110
T	2	1110
N	2	110
X	5	10
E	6	01
U	7	00
Velikost kodu(bit)	suma=24	60

Ikdyž tato věta ve většině případů platí, není to pravidlo, které by platilo obecně pro každý případ.

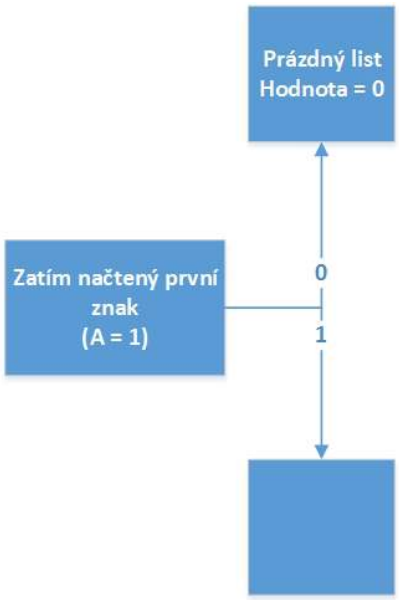
9.2. Adaptivní Huffmanovo kódování

Metoda vychází z toho, že zná četnost výskytu obsažených znaků v řetezci. Bohužel toto je málokdy známé, aniž by byly data čtená dvakrát. Jednou pro zjištění četností a podruhé, aby byla provedena komprese, kdy se mezi průchody vytvoří „strom“. Tato metoda je příliš pomalá a označuje se semiadaptivní.

V praxi se používá metoda adaptivní, která je základem pro UNIX. Metoda řeší problém s dvojím průchodem tak, že si na začátku vytvoří prázdný Huffmanův strom a při prvním průchodu do něj bude zapisovat symboly a v průběhu zpracovávání jej bude upravovat, než budou data zpracovaná - komprimovaná. Jelikož se kód v průběhu může měnit, je potřeba aby se v opačném případě měnil i dekompresor, aby bylo možné opět zakódované data rozkódovat. Nejlépe by znělo označení, kdy se komprese a dekomprese zrcadlí, ale také může pracovat provázaně (lockstep), pracují v každém případě synchronně.

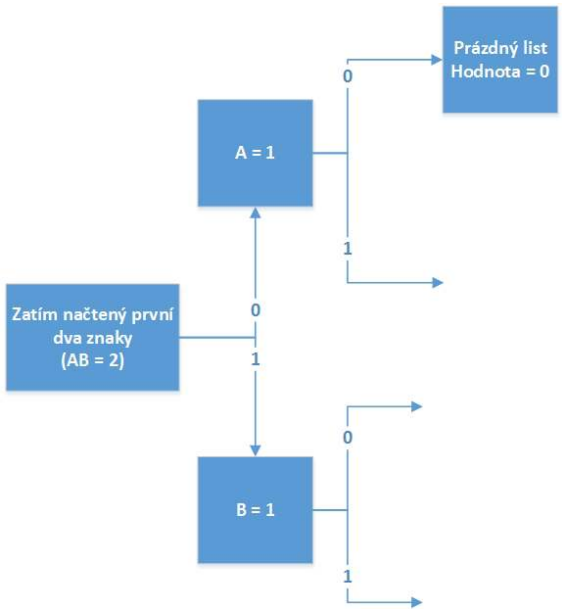
Největší problém nastává při zjištění, jestli položka, která se dostala do dekompresoru je nekomprimovaná nebo ne (symbol je většinou v 8-bitovém ASCII kódu). Řeší se tak, že před každým nekomprimovaným symbolem bude vložen kód změny (escape code) s proměnnou délkou. Tím pádem bude lehce určitelné, že následujících 8 bitů je symbol v ASCII kódu, který není zatím přiřazen.[5]

Nejedná se o jediný problém, který při vytváření nastává. Jelikož kód změny nesmí být shodný s proměnnou délkou již použitých symbolů. Jelikož se začíná s prázdným stromem, tak je potřeba při každé změně také upravit výsledný kód. Způsob, jak se vyvarovat problémům je takový, že se vždy přiřadí ke stromu prázdný (nulový) list 9.4, který bude mít nulovou četnost. Tento list označujeme kód změny, kerý předchází již zmíněný nový nekomprimovaný kód.



Obrázek 9.4: Adaptivní Huffmanovo kódování - vytvoření prázdného listu.

Jak se strom neustále upravuje a provádí se změny, tak i tento kód se mění (vždy na stranu kde je hodnota větve 0). 9.4



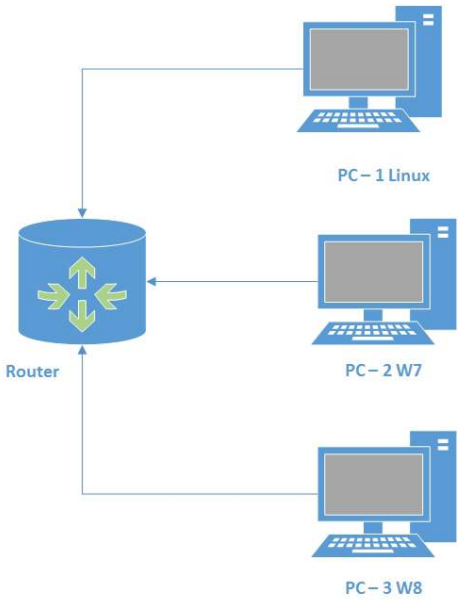
Obrázek 9.5: Adaptivní Huffmanovo kódování - část komprese s prázdným listem.

Jakmile se přidají další proměnné do stromu, je potřeba opět přiřadit prádný list s hodnotou „0“ na pravou větev diagramu. 9.5

10. LINUX - Ubuntu

Simulace DoS útoku bude provedena na operačním systémem Linux Ubuntu, ke stažení ZDE. Bude vytvořena malá síť, která bude obsahovat 3 počítačové stanice.10.1 Na jedné bude již zmíněný Linux a na dvou Windows 7 a 8. Stanice budou ve stejné vnitřní síti, aby bylo možné provést DoS útok, jelikož je možné, že poskytovatelé budou mít mechanismus na detekci a případnou obranu proti takovému útoku, ikdyž se jedná o slabý útok.

Na stanici PC-1 bude nainstalován Linux, pomocí kterého budeme posílat nepřetržitě velké množství dotazů na PC-2 a na kontrolu si můžeme z PC-3 posílat Ping z příkazového řádku a sledovat jak je stanice přístupná během útoku.



Obrázek 10.1: Síťové zapojení koncových stanic

PC-1 Linux Po instalaci je potřeba systém aktualizovat a stáhnout software, který nám umožní provádět nastavení a Hping v síti. V první řade využijeme příkazy:

sudo apt-get update

sudo apt-get upgrade

Jedná se o příkazy pod superuživatelem, který má právo provádět instalace, update a úpravy systému. K tomuto je potřeba i heslo. Tímto se nám stáhnou nové aktualizace a nainstalují a po ukončení instalací je potřeba restart systému.

Další z příkazů bude potřeba použít

sudo apt-get install hping3

Tímto příkazem se nainstalují balíčky pro rozšíření Hping. Pokud nefunguje tento příkaz, je možné stáhnout přímo balíček z internetu a ten posléze nainstalovat.

Nástroj je možné použít také na zjištění a otestování sítě v těchto oblastech:

- Trasování, ping
- Kontrola nečinnosti, nyní implementováno s nmap
- Pravidla firewallu
- Testování TCP/IP
- Síťové vyhledávání

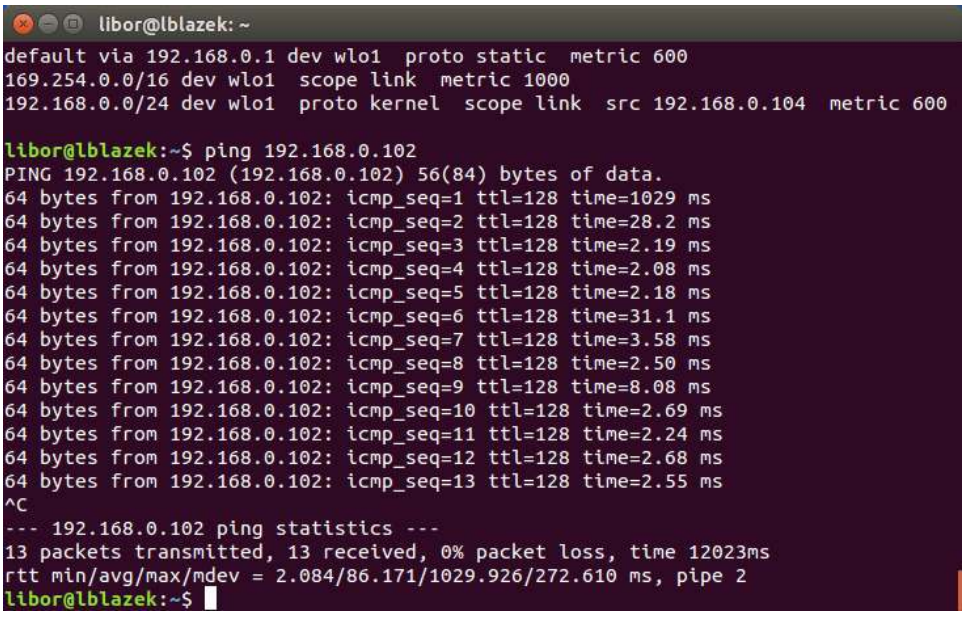
Abychom mohli sledovat využití komunikace v síti, bude potřeba stáhnout ještě program Wireshark. Použije se stejný příkaz jak v předchozím případě, ale na konci bude "wireshark".

Na stanicích si zjistíme IP adersu pomocí příkazového řádku (cmd) tak, že vypíšeme ipconfig. Nyní jsou zjištěny veškeré informace k tomu, aby mohla být provedena simulace. Posílání paketů bude probíhat z PC – 1 na PC – 2, bude se také posílat pro kontrolu dostupnosti i „ping“ jak z PC – 1 tak i z PC – 3.

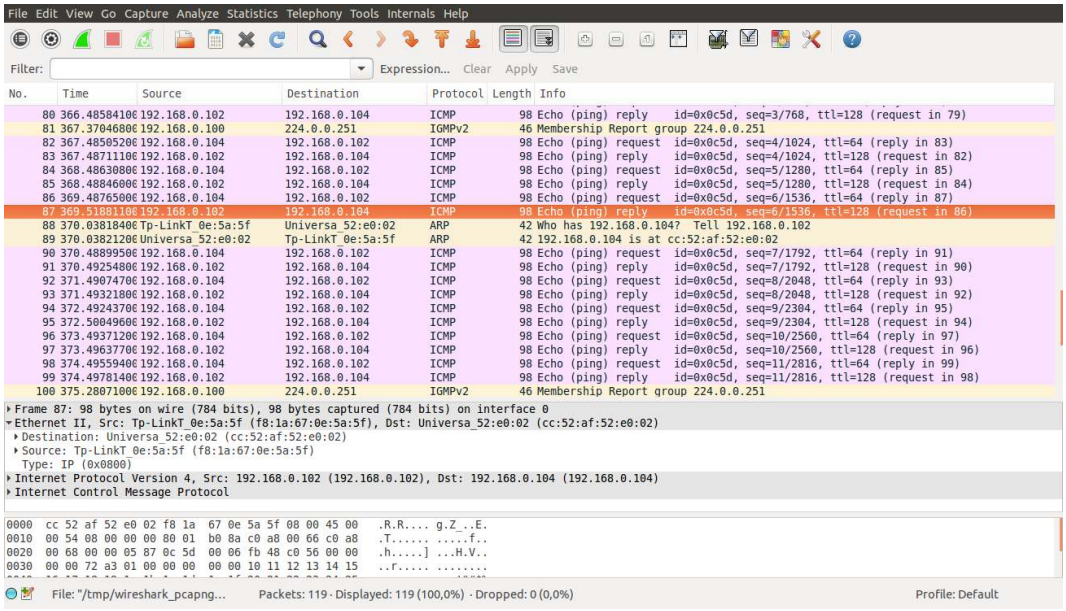
10.1. Simulace DoS

Spustíme si stanici s nainstalovaným operačním systémem Linux Ubuntu a spustíme Wireshark pomocí příkazu *sudo wireshark* v terminálu (Ctrl + Alt + T), pokud bychom nepoužili sudo nemusel by program pracovat korektně. Z nabídky vybereme aktivní připojení, v našem případě „wlo 1“ a spustíme sledování. Terminál nesmíme zavírat, jinak se vypne program.

Otevře se další terminál, na kterém bude puštěný příkaz "ping"na IP adresu PC - 2 (192.168.0.102)10.2 a pomocí wiresharku sledujeme komunikaci.10.3



Obrázek 10.2: Zjištění dostupnosti PC pomocí příkazu PING na IP adresu.

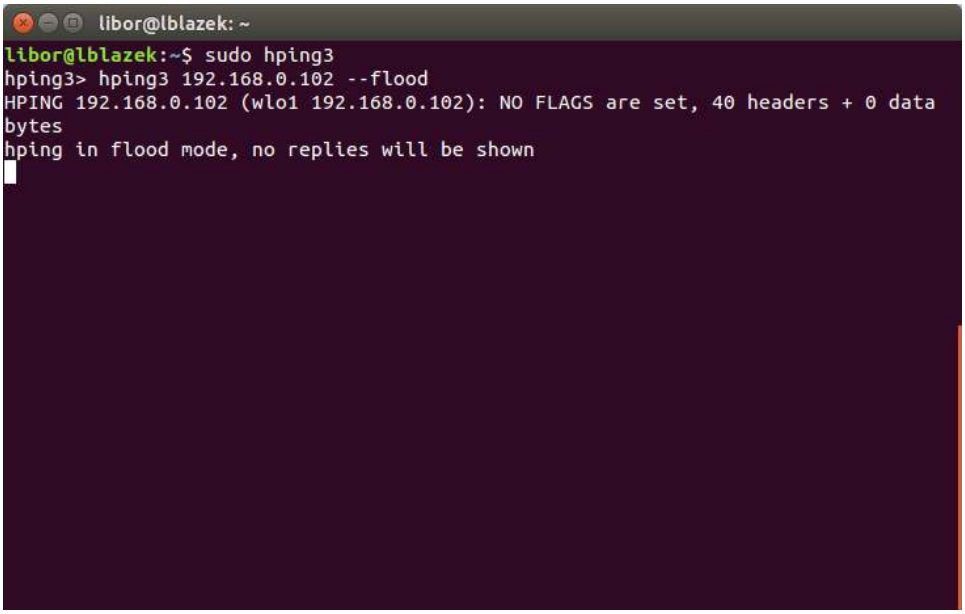


Obrázek 10.3: Zachycení dotazu PING na koncovou stanici pomocí programu Wireshark.

Podle zachycené komunikace můžeme zjistit, že příkazem „ping“ posíláme dotaz na cílovou (destination) adresu 192.168.0.104 „ECHO (ping) request“ a zpátky je posílaná odpověď „reply“ na adresu 192.168.0.102. Použitý protokol je „ICMP“, který slouží ke zjištění dostupnosti prvků v síti, jestli jsou služby v provozu a zasílání chybových zpráv.

Nyní je ověřená komunikace mezi zařízeními v síti a může se přejít k zahlcení stanice. K tomu je potřeba otevřít další terminál, na kterém se spustí „hping3“ a k němu příslušný parametr.

Do rozhraní hping3 se musí přes příkaz sudo hping3, poté se terminál přepne do indexu ”hping3>”. Nyní je možné použít příkaz na danou IP adresu a parametr.10.4



Obrázek 10.4: Zahlcení PC stanice pomocí příkazu hping3.

Je potřeba k tomuto příkazu použít parametr - -flood, který slouží k zahlcení komunikace. Mužeme si všimnout, že se posílá velké množství dotazů (sleduje se přes wireshark). Terminál informuje o tom, že byl použit flood mode a nebudou zobrazené žádné informace. Ty se zobrazí, jakmile bude přenos ukončen (CTRL + C).

Mohou se použít další parametry a nastavení:

- **–d 120** = velikost každého paketu
- **–s** = odeslání pouze SYN paketu
- **–p 80** = nastavení cílového portu

Pro větší detaily provozu mohou být hodinové intervaly dále rozděleny na minutové úseky. Rozdělení bylo provedeno pomocí příkazu:

editcap -i 60 vstupni.pcap vystupni.pcap

Číslice 60 v příkazu značí, že soubory jsou děleny po 60 vteřinách. V případě potřeby jiného časového intervalu lze toto číslo změnit na požadovanou hodnotu.

11. Visual Studio

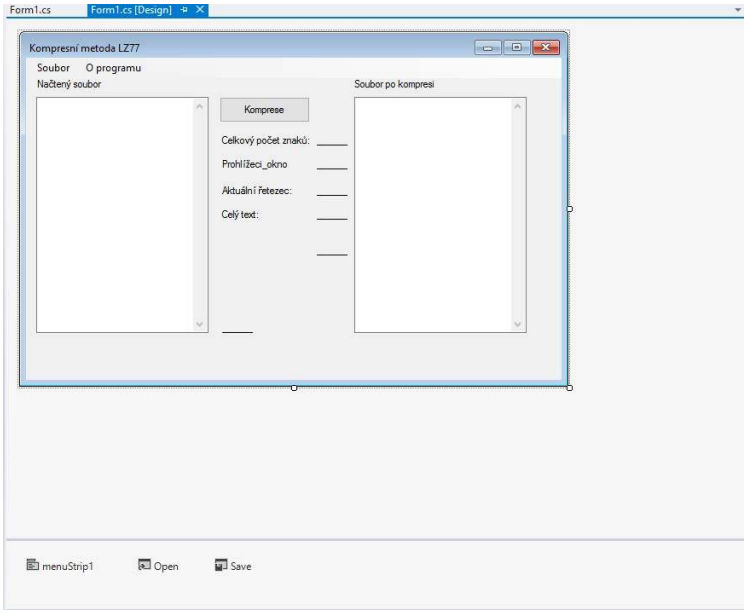
Program je vytvořen firmou Microsoft, který slouží k vytváření konzolových aplikací, Windonws forms aplikací (jedná se o klasický program) a dalších projektů. V MVS (Microsoft Visual Studio) je možné psát v několika programovacích jazycích, jako je C, Java a dalších, která jsou možná stáhnout přes webové stránky, nebo rozšířit v nastavení. Můžeme najít několik verzí jako Express nebo Enterprise s označením roku 2012 nebo 2015. Tento software je volně ke stažení (záleží která verze je instalována), nebo v rámci MSDNAA na studenstské využití s případným klíčem na licenci. Na tvorbu je vybrána konkrétní verze: Visual Studio 2015 ke stažení: ZDE a programovací jazyk je vybrán C#.

Je potřeba zdůraznit, že celý kód bude přiložen na CD. Popsány budou jenom důležité části a funkce sloužící k funkčnosti programu.

11.1. LZ77

Jak bylo již zmíněno, jako programovací jazyk bude použit C#. Vytvářet se bude aplikace stylu „Forms“. Lze si jej vybrat při vytváření nového projektu, kde je potřeba si v tomto případě vybrat Visual C a následně „Windows Forms Application“. Dále je možnost si projekt pojmenovat nebo jej umístit na konkrétní cestu, kde bude uložen.Na spuštění programu bude sloužit soubor uložen ve složce **bin** a následně **Debug**.

V první řadě je potřeba vytvořit vizuální stránku programu, kde bude zobrazena základní kostra programu, veškeré popisky, okna, tlačítka atd. V pozdějších částech se mohou dodělat méně důležité prvky. Zde bude potřeba vytvořit dvě okna. Jedno okno bude sloužit k vložení původního textu (nebo načtení se souboru .txt) a další okno bude na vypsání kompresního výstupu. Aby bylo možné provést tento úkon je potřeba vložit tlačítko. K načtení souboru bude vytvořena horní lišta, která bude obsahovat jednoduché menu.11.1



Obrázek 11.1: Grafický návrh pro kompresní metodu LZ77

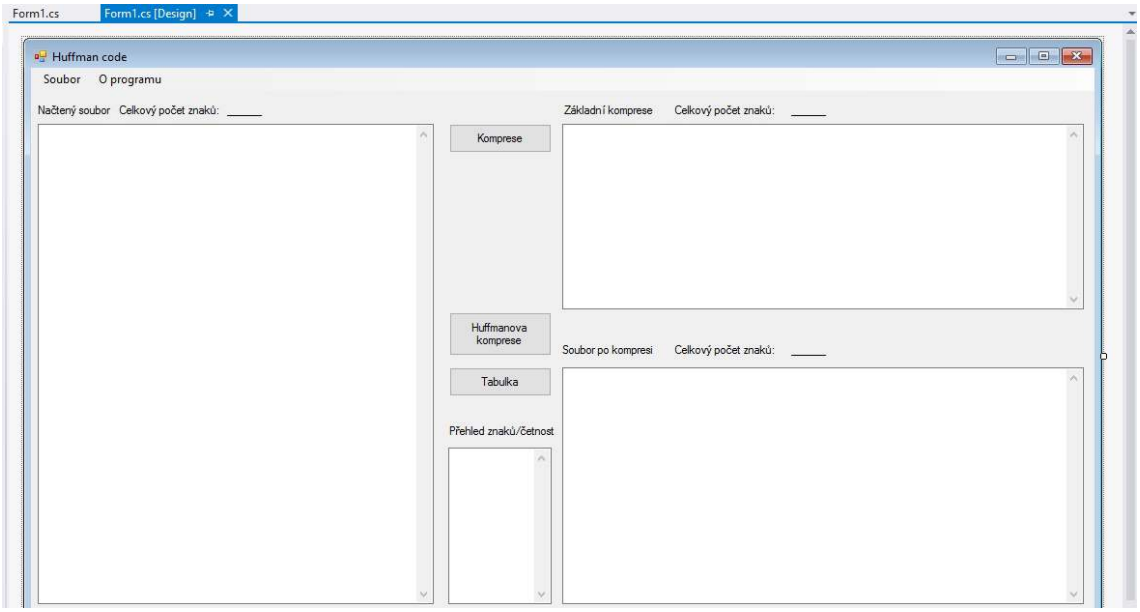
Pokud je potřeba pro zobrazení mezikroku, nebo nějakých výsledků z důvodu kontroly, je možné si vložit „label“ který může být následně zneviditelněn a v programu se nebude zobrazovat.

11.2. Huffmanovo kódování

Tento návrh bude muset být odlišný než u LZ77. Bude potřeba více výpočetních úkonů, talčítek a zobrazovacích polí.

Jelikož bude nutné vytvořit i Huffmanův strom, bude použit i další zobrazovací okno, které se bude zobrazovat po stisknutí tlačítka.

základní struktura, jako je horní lišta, textbox na vstup a výstupy zůstane stejná.11.2



Obrázek 11.2: Grafický návrh pro Huffmanovu kompresní metodu.

12. Wireshark

Je potřeba získat zdrojová data k následné kompresi. Vytvoří se dva soubory z programu Wireshark. Jeden soubor bude tvořen běžnou komunikací v síti a v druhém souboru bude zaznamenán útok, pomocí druhé stanice. Aby se komprese projevila, tento útok bude muset probíhat po určitou dobu.

Po ukončení sběru dat si můžeme tyto informace opět zobrazit. Vzniklé soubory jsou ve formátu pcap. Wireshark zaznamenává spoustu informací, které budou pro tuto kompresi nepotřebné, a bude nutné je odstranit. Nejdůležitější bude získání IP adres, ze kterých přichází komunikace na danou stanici (server). Pro porovnání příchozí komunikace na stanici a vyhodnocení, jestli na stanici bylo posíláno neúměrné množství dotazů za účelem omezení dostupnosti, je nutné vytvořit také soubor, kde bude uložen běžný provoz. Je nutné zjistit jakou kompresi bude mít běžná komunikace, aby bylo možné následně porovnat kompresi, kdy bude nasimulován útok pomocí dotazů na IP adresu stanice. Tento záznam se vytvoří tak, že se na stanici spustí program Wireshark a provede se případné nastavení. Je nutné přiřadit na sledování danou síťovou kartu a program spustit. Téměř okamžitě se začne v informačním okně zobrazovat aktuální komunikace. Nechají se zaznamenávat data po určitou dobu. Uživatel v tomto čase není nějak omezen z běžného provozu.

12.1. Pcap

Pcap je formát souboru spjatý převážně s programem Wireshark, který dokáže zachytávat komunikaci v síti. Díky těmto souborům a programem lze takto uložená data otevírat a analyzovat datové přenosy. Název vznikl z odvození dvou slov – Packet capturing, v překladu, zachytávání paketů. Pro Windows systémy se implementuje WinPcap a pro Unixové jako libpcap. V těchto uložených formátech jsou data, která obsahují informace o komunikaci již od linkové vrstvy referenčního ISO/OSI modelu.

Pokud by byla potřeba z takto uložených dat získat pouze informace z daného období, je možné použít příkaz „editcap“. Díky parametrům, které se následně použijí, se mohou upravit vstupní data a vytvořit nové s námi nadefinovanými parametry.

12.2. Reálný provoz v síti

V této části se provede útok, který byl již simulován v předchozí kapitole. Vytvoří se stejná síť, která bude tvořena třemi stanicemi. Na jedné stanici bude spuštěn Wireshar (PC Windows 7), další stanice bude mít operační systém Linux, ze kterého bude simulován útok a poslední bude sloužit na kontrolu, jestli je útok prováděn (bude se odesílat PING, aby se zjistila dostupnost stanice). Je potřeba zaznamenat určitý počet dat, proto bude potřeba nechat stanice pracovat v běžném provozu kolem 1 min (bude vytvořeno několik minutových záznamů). Tato data se uloží následně do souboru „provoz_1_min_X.pcap“ a bude z něj extrahována potřebná data, tedy IP adresy.

Jakmile se extrahují IP adresy ze souboru, provede se následná komprese pomocí Huffmanovi metody. Zjistí se celkový počet prvků, jejich četnost a následně se převedou do binárního čísla.

Nyní se zpřístupnila možnost provést kódování pomocí Huffmanova algoritmu. V případě většího počtu záznamů, tento proces, ale i předešlý, může trvat určitou dobu, než se nahradí veškerá data na vstupu.

Sběr dat může proběhnout v jakémkoliv období. V tomto případě bude časový rámec obsahovat 5 minut. Spustí se program "Wireshark", vybere se síťový prvek bude využit. Následně se spustí aplikace a nadále se bude moct pracovat na stanici stejně, jako při běžném užívání.

V programu můžeme sledovat kolik bylo zachyceno paketů, buď podle popisku nebo ve spodní liště.

Po ukončení časového rámce, je potřeba zastavit zachytávání komunikace a následně jej uložit do souboru. Je potřeba, aby byl zvolen soubor s koncovkou *.pcap, ze kterého se budou následně extrahovat IP adresy.

12.3. Reálný provoz v síti s útokem na koncovou stanici

V této části je potřeba provést cílený útok pomocí DoS. K tomu bude sloužit již zmíněná stanice s operačním systémem Linux. Jak bylo již v dřívější kapitole zmíněno, je nutné v terminálu pomocí danného příkazu provést posílání dotazů na dannou IP adresu.

Nechme tedy opět zapnout na stanici (na kterou se bude útočit) "Wireshark". Nemusí se hned provádět útok. Nějakou dobu můžeme sledovat provoz a po určitém čase provést příkazem "hping3"zasílání dotazů (z PC-1).

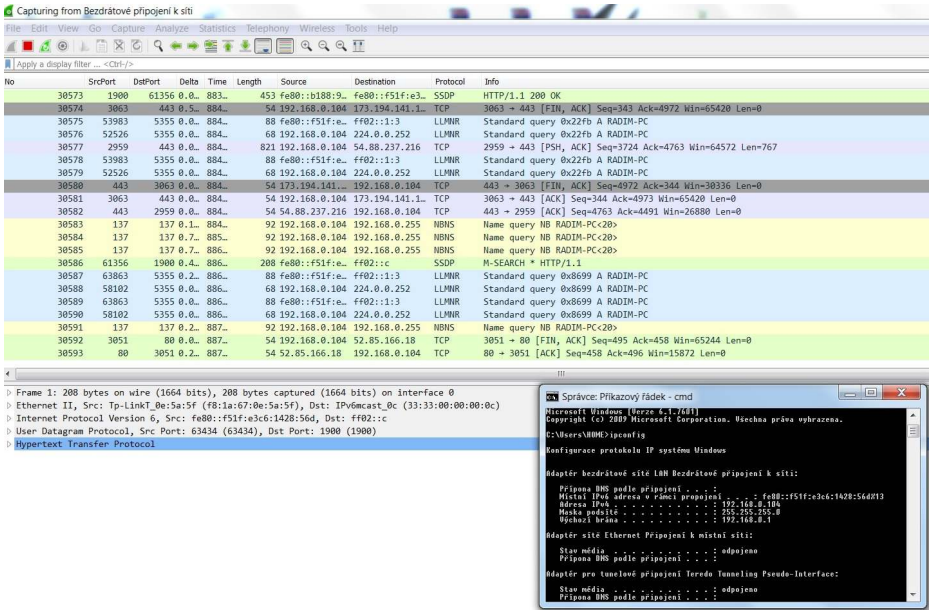
Takto zahltit stanici můžeme na libovolně dlouhou dobu. Avšak v tomto případě můžeme nechat útok běžet přibližně několik vteřin. Poté můžeme útok přerušit a po nějakém čase opět spustit. Tímto způsobem se nám zaznamená mnohonásobně více IP adres, kterou má útočící stanice.

Před ukončením programu wireshark můžeme také zjistit, že došlo podstatně více dotazů na stanici (počet paketů). Z tohoto údaje můžeme odhadnout, kolik bude na vstupu celkem znaků. Při běžném provozu se počet pohyboval kolem stovek tisíc dotazů. V kompresi budeme používat pouze IPv4 adresy. Ne všechny pakety budou zobrazeny ve tvaru námi požadovaném, proto bude IP adres vždy méně, než zachycených paketů. Je dobré si také uvědomit, že adresa je tvořena celkem 11 znaky. Při větším počtu by nemusel námi vytvořený "textbox"pojmout všechny znaky.

Z tohoto důvodu je nutné také mít připravený "TextBox", který dokáže takové množství dat zobrazit.

13. Výsledky komprese - běžný provoz

Během standartního provozu v síti, můžeme sledovat komunikaci probíhající s koncovou stanící a ostatními prvky v síti a na internetu.13.1 Na obrázku je zobrazena ukázka, jak může taková komunikace vypadat v programu Wireshark a v příkazovém řádku jsou zobrazeny informace ohledně nastavění síťové adresy koncové stanice, na kterou se následně bude provádět i útok pomocí DoS. Veškeré ukázky komprese a grafy komunikace jsou zobrazeny v kapitole 19.

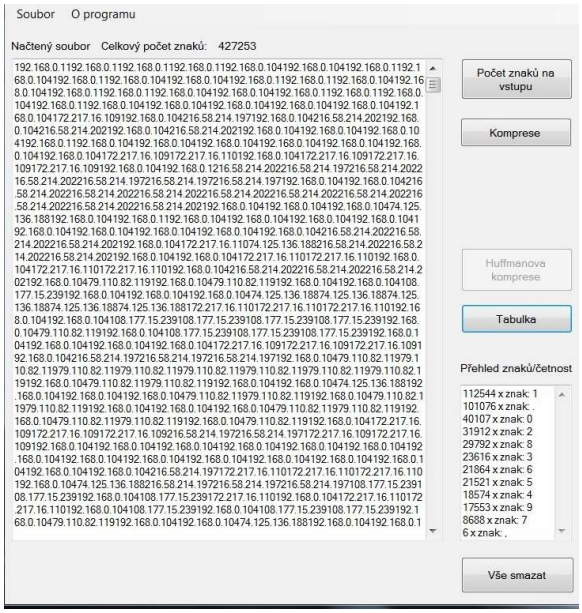


Obrázek 13.1: Zobrazení komunikace pomocí programu Wireshark.

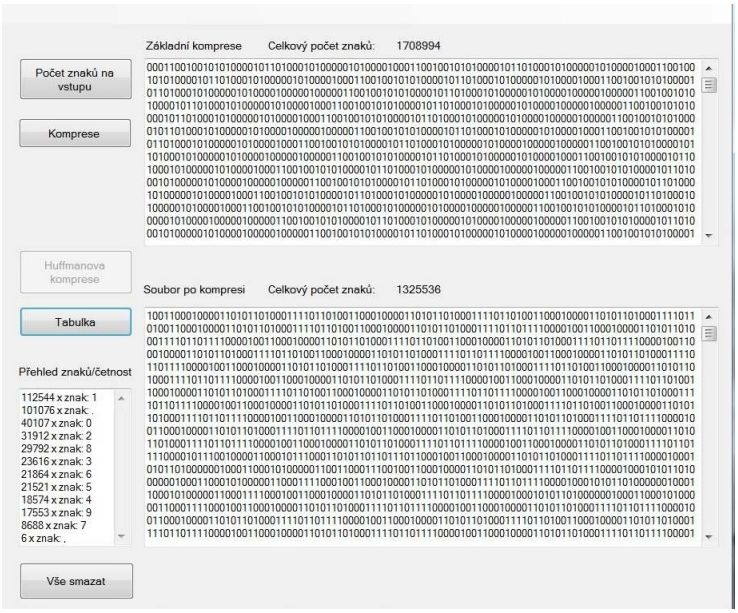
Tato komunikace se následně uložila a pomocí příkazu($tshark - r < vstup > -T fields - eip.src < vystup >$) se extrahovala potřebná data. Podle změny parametru, můžeme získat libovolná data, která jsou v nabídce a je možné je získat. V tomto případě bude potřeba získat IP adresy z položky "source"a převést je do textového souboru.

Z ohledem na velikost souboru a počtu znaků, je příhodnější otevřít raději textový soubor a veškeré údaje zkopírovat přímo do textboxu pro vstup. Může se stát, že větší soubory nepůjdou otevřít z důvodu velkého množství dat, řádově od 1.5MB, a následného přetížení programu.13.2 Je lepší buď zmenšit soubory na více částí, nebo data přímo nakopírovat do programu.

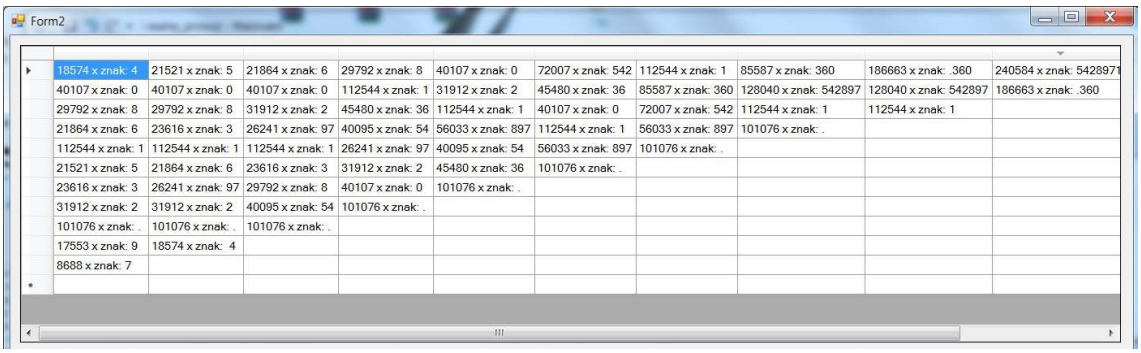
Aby byla provedena Huffmanova komprese, je nutné napřed zjistit četnost a přehled znaků. Poté se zpřístupní tlačítko na provedení Huffmanovi komprese. Je dobré provést i kompresní metodu, která nahradí každý znak binární hodnotou, kdy je velikost stejná pro každý znak, aby se porovnaly dvě různé metody.13.3



Obrázek 13.2: Načtená vstupní data (i četnost) a připravená na kompresi.



Obrázek 13.3: Výstupní data po kompresi: klasická (nahoře) a Huffmanova (dole) metoda.



18574 x znak: 4	21521 x znak: 5	21864 x znak: 6	29792 x znak: 8	40107 x znak: 0	72007 x znak: 542	112544 x znak: 1	85587 x znak: 360	186663 x znak: .360	240584 x znak: 5428971
40107 x znak: 0	40107 x znak: 0	40107 x znak: 0	112544 x znak: 1	31912 x znak: 2	45480 x znak: 36	85587 x znak: 360	128040 x znak: 542897	128040 x znak: 542897	186663 x znak: .360
29792 x znak: 8	29792 x znak: 8	31912 x znak: 2	45480 x znak: 36	112544 x znak: 1	40107 x znak: 0	72007 x znak: 542	112544 x znak: 1	112544 x znak: 1	
21864 x znak: 6	23616 x znak: 3	26241 x znak: 97	40095 x znak: 54	56033 x znak: 897	112544 x znak: 1	56033 x znak: 897	101076 x znak: .		
112544 x znak: 1	112544 x znak: 1	112544 x znak: 1	26241 x znak: 97	40095 x znak: 54	56033 x znak: 897	101076 x znak: .			
21521 x znak: 5	21864 x znak: 6	23616 x znak: 3	31912 x znak: 2	45480 x znak: 36	101076 x znak: .				
23616 x znak: 3	26241 x znak: 97	29792 x znak: 8	40107 x znak: 0	101076 x znak: .					
31912 x znak: 2	31912 x znak: 2	40095 x znak: 54	101076 x znak: .						
101076 x znak: .	101076 x znak: .	101076 x znak: .							
17553 x znak: 9	18574 x znak: 4								
8688 x znak: 7									

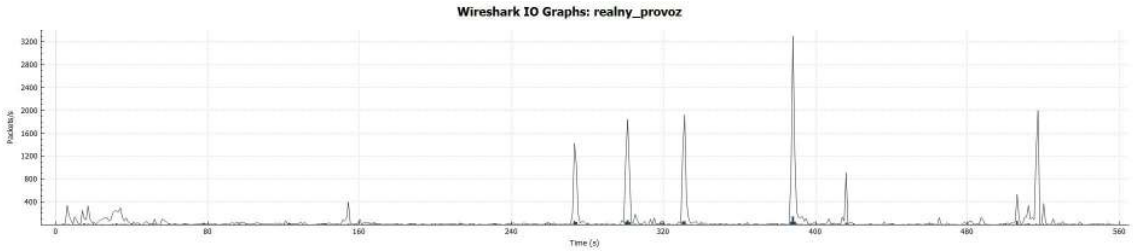
Obrázek 13.4: Zobrazení tabulky ve které se vytvářel Huffmanův strom.

V obrázku 13.4, jsou zobrazeny kroky, které probíhaly díky kompresi. Pro kontrolu můžeme použít méně obsáhlé řetězce, nebo porovnat seče všech prvků v poslední buňce, která obsahuje celkovou délkou vstupního řetězce. Pokud se rovnají je správně sečteno.

Jestliže se podíváme na první sloupeček v tabulce, zjistíme, že nejméně znaků v řetězci jsou znaky tvořeny číslem 9 a 7. Jelikože Huffmanova komprese pracuje s vždy nejmenší četností, tak se tyto dva znaky (jejich četnost) sečtou a vytvoří se i nové označení (pro kontrolu 97). Následně se sloupeček opět seřadil podle velikosti a vzali se dva nejnižší prvky (5 a 4) a sečetli. Tímto způsobem se pokračovalo až do konce tabulky.13.4

Vždy, když se sčítala dvě čísla a proměnné, bylo přidáno v pozadí programu ke každému znaku hodnota 0 nebo 1, aby se vytvořil výstup pro každý znak, který tímto bude poté nahrazen.

Můžeme si všimnout rozdílu v kompresních metodách a jejich velikosti v počtu znaků.13.3



Obrázek 13.5: Grafické zobrazení síťového provozu při běžném provozu.

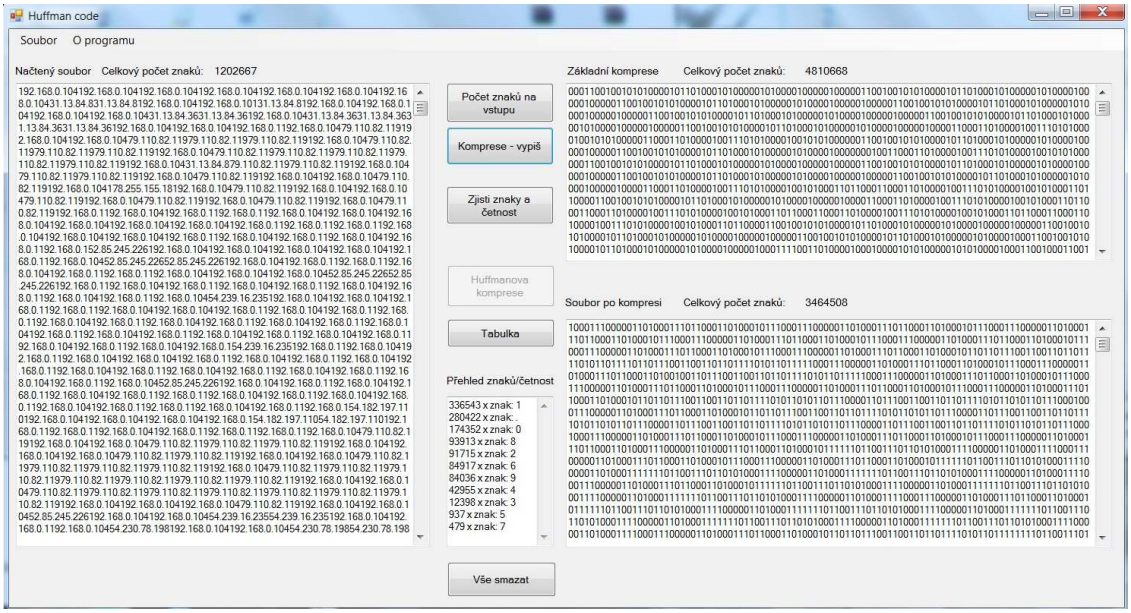
Pokud by jsme otevřeli již uložený soubor z programu Wireshark, můžeme si nechat zobrazit graficky průběh komunikace.13.5 Podle grafu můžeme zjistit, že se počet paketů dostal k maximální hodnotě nad 3 200 paketů při větším vytížení. V další části bude ukázán rozdíl při zahlcení síťového provozu.

14. Výsledky komprese - zahlcení síťového provozu

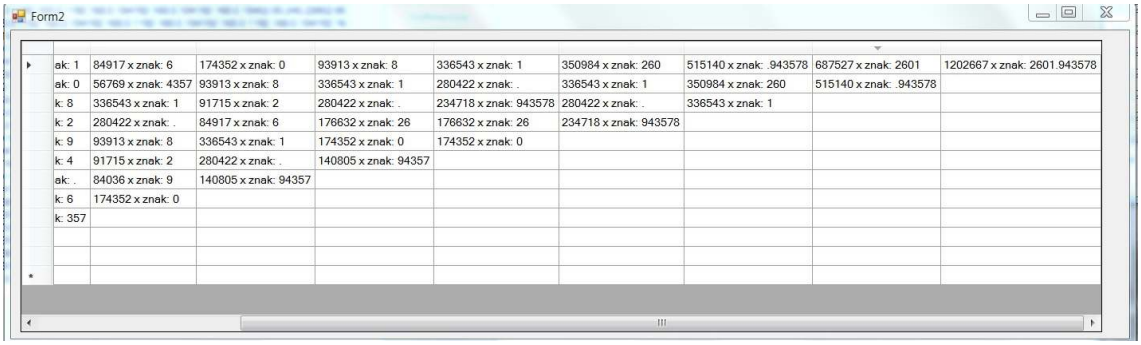
Bude ukázán jeden případ a další data budou zobrazena v kapitole 19.

Jak již bylo několikrát zmíněno, je potřeba brát ohled na celkový počet znaků na vstupním řetězci a také také počítat s tím, že některé úkony mohou trvat i několik minut (i desítek). Po extrahování IP adres a zkopírování je do vstupu programu můžeme zjistit, že za stejné období je již počet znaků přes jeden milion.

Je zde potřeba opět zjistit znaky a jejich četnost. Následně se může provést komprese obou metod.14.1



Obrázek 14.1: Komprese IP adres z DoS útoku pomocí Huffmanovi metody.



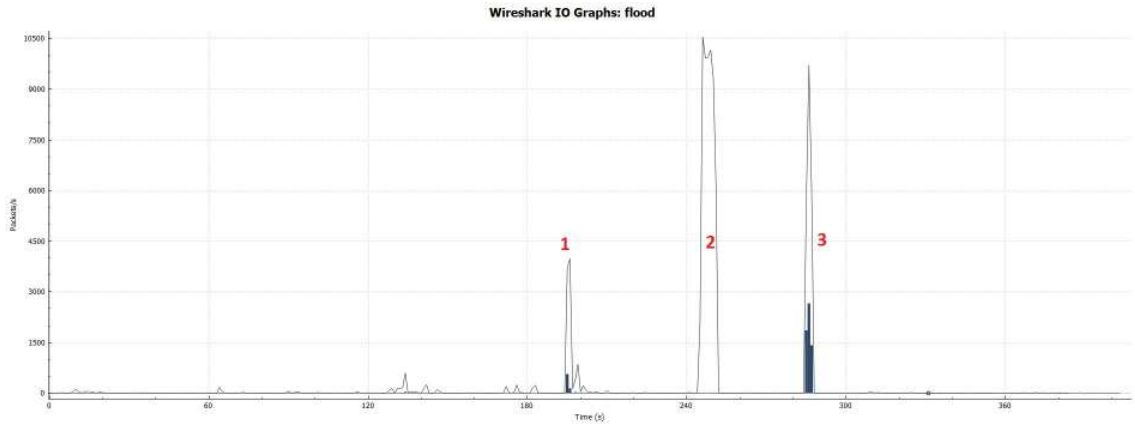
ak: 1	84917 x znak: 6	174352 x znak: 0	93913 x znak: 8	336543 x znak: 1	350984 x znak: 260	515140 x znak: .943578	687527 x znak: 2601	1202667 x znak: 2601.943578
ak: 0	56769 x znak: 4357	93913 x znak: 8	336543 x znak: 1	280422 x znak: .	336543 x znak: 1	350984 x znak: 260	515140 x znak: .943578	
k: 8	336543 x znak: 1	91715 x znak: 2	280422 x znak: .	234718 x znak: 943578	280422 x znak: .	336543 x znak: 1		
k: 2	280422 x znak: .	84917 x znak: 6	176632 x znak: 26	176632 x znak: 26	234718 x znak: 943578			
k: 9	93913 x znak: 8	336543 x znak: 1	174352 x znak: 0	174352 x znak: 0				
k: 4	91715 x znak: 2	280422 x znak: .	140805 x znak: 94357					
ak: .	84036 x znak: 9	140805 x znak: 94357						
k: 6	174352 x znak: 0							
k: 357								

Obrázek 14.2: Zobrazení tabulky při DoS útoku, ve které se vytvářel Huffmanův strom.

V tabulce 14.2 si můžeme opět ověřit správnost funkce podle poslední buňky, kde počet souhlasí s celkovou délkou vstupního řetězce a jsou v něm obsaženy všechny znaky.

Pokud bychom chtěli sledovat výsledek graficky, načteme uložený soubor z wiresharku a zobrazíme si komunikaci v grafu.

Podle grafu 14.3, můžeme vidět komunikaci, kdy se špičky na začátku dostávali do hodnot kolem 500 paketů při běžném provozu s následnou odchylkou do 3 500, kde mohla být spuštěna aplikace, načítat video na webové stránce, nebo jiné data.



Obrázek 14.3: Grafické zobrazení síťového provozu při DoS útoku.

Následně byl poslán "hping3"z jiné koncové stanice v síti a můžeme sledovat, jak se počet paketů zvětšil řádově na 10 000. Po pár vteřinách byl tento útok ukončen a po chvíli zase zopakován.

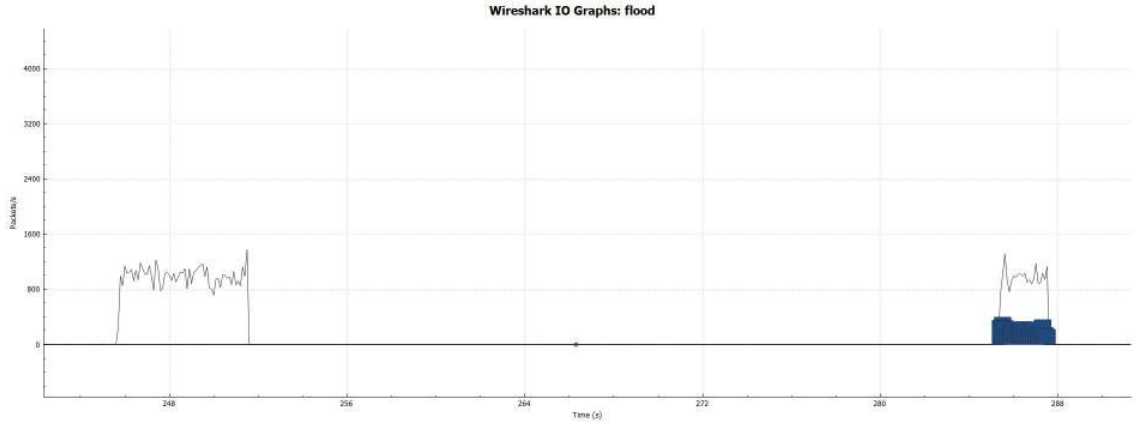
Odchylky vytvořené v grafu:

1 Jedná se o klasický provoz, kdy bylo spuštěno na ukázku video v prohlížeči, aby se vytvořil náročnější přenos dat.

2 První posílání DoS na koncovou stanici. Lze porovnat s přenosem videa, kdy útok posílá několikánásobně více dotazů a dat, než video.

3 Druhé zasílání DoS.

Pokud bychom si grafy přiblížili 14.4, můžeme si všimnou, že jsou zobrazené ještě další data (modrý sloupcový graf). Většinou jsou zobrazeny při útocích, nebo si můžeme všimnou i u spuštění vídea14.3 (1). Jedná se převážně o zahozené pakety, které nebyly zpracovány. Více grafů je zobrazeno v příloze, na konci práce. Změní se také počet paketů za uplynulou dobu. Zde můžeme sledovat, že maximální počty paketů se pohybovaly od 800 do 1 400 paketů za časový údaj.

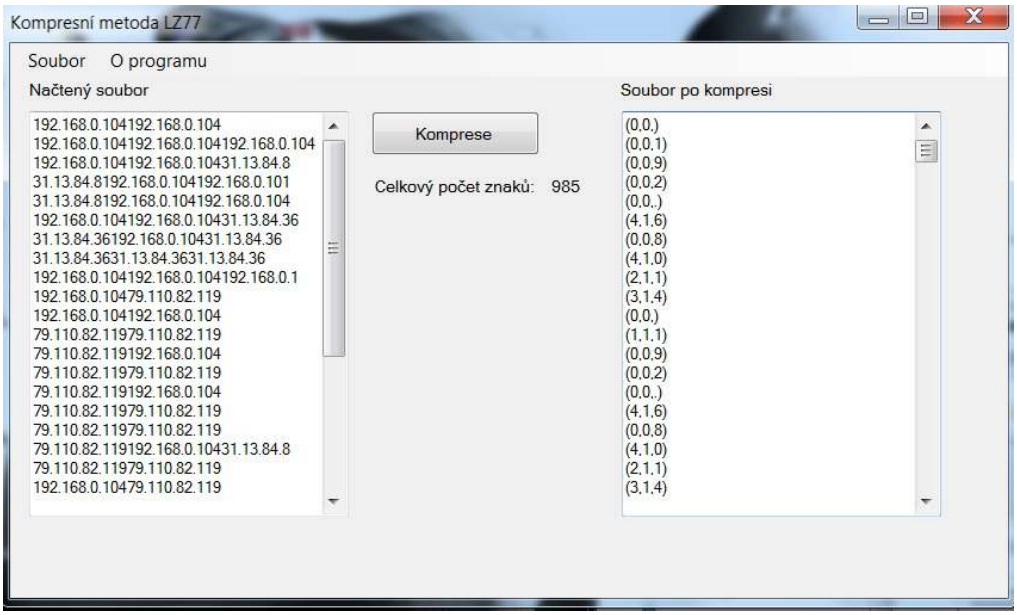


Obrázek 14.4: Detail provozu při DoS útoku.

15. Výsledky kompresní metody LZ77

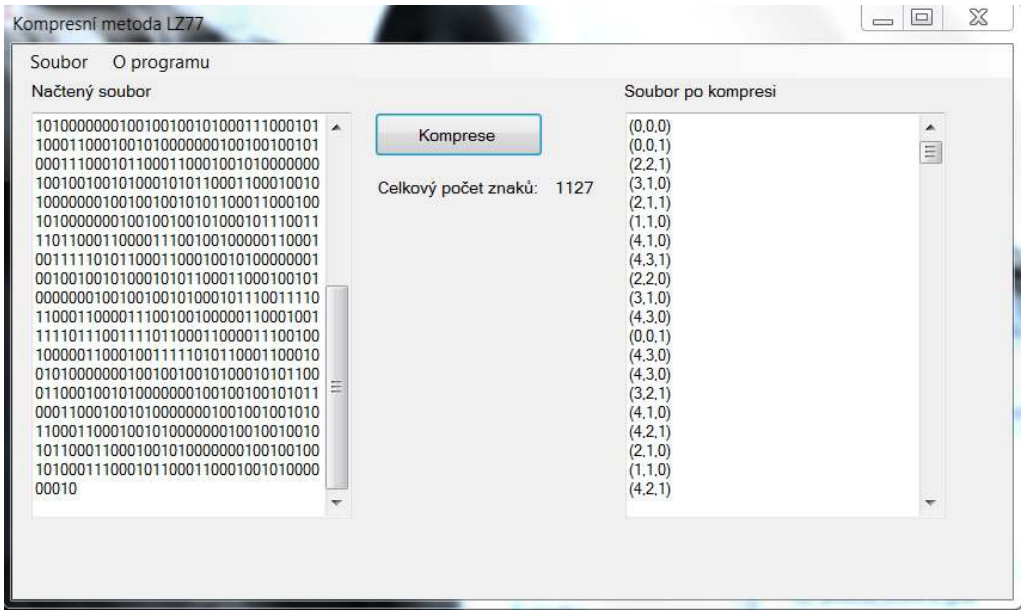
K testování kompresní metody pole LZ77 budou sloužit stejné zdrojové soubory, jako u předchozí metody. Pro přehlednost bude však stačit, když použijeme nějakou část těchto dat.

Na výstupu je uveden každý výsledek zvlášť, na nový řádek, aby byla komprese přehlednější. Můžeme si zkopírovat část dat při komunikaci v běžném provozu. Jelikož je vždy na výstupu vytvořen nový řádek, musí se počítat s obsáhlejším výsledkem, a proto je nutné nastavit také velikost paměti v textboxu. Použijí se nyní IP adresy, které byly získáný při zachytávání komunikace. 15.1 V levé části můžeme vidět řetězec IP adres a v pravém boxu je zobrazen výsledek po kompresi. Je také dobré zaměřit se na hodnoty v závorce, hlavně prostřední, která představuje počet shodných znaků. Zde je ve výsledku převážně shodný jeden znak, na jiných pozicích, tudíž se komprese nebude pohybovat ve velkých změnách.15.1



Obrázek 15.1: Komprese IP adres pomocí metody LZ77.

V dalším případě je možné použít obě metody dohromady. Je možné provést prvně kompresi pomocí Huffmanovi metody, kdy na výstupu budou pouze znaky 1 a 0. Na tomto výstupu se může následně provést metoda LZ77. Na ukázku použijeme kratší vstupní řetězec. Již na prvních řádcích,15.2 výstupu metody LZ77, si můžeme všimnout, že se vytváří komprese, kdy hodnoty "j"mají jinou hodnotu, než v předešlých případech. Je to způsobeno tím, že jsou na vstupu pouze dvě hodnoty, které se opakují.



Obrázek 15.2: Komprese dat z Huffmanovi metody a následně LZ77.

16. Závěr a výsledky kompresních metod

Bakalářská práce se zabývala možnými útoky na koncová zařízení. Dále byly probrány některé z kompresních metod, jako je například LZ77, LZW a Huffmanova komprese. Ukázali jsme si i příklady, jak tyto metody pracují při kompresi a následné dekompresi.

Bylo nutné vytvořit program, který na vstupu, což mohou být data ze souboru nebo ručně vepsaná, vytvoří na výstupu data po dané kompresi.

Byly vytvořeny dva programy, na kterých je provedena ukázka komprese a to pomocí LZ77 a Huffmanovi kompresní metody.

Byly použité krátké vstupní řetězce, aby bylo možné provést rychlou kompresi a prověřit jejich funkčnost. Dále byla vybrána metoda pomocí Huffmanovi komprese na další testování.

Bylo potřeba zjistit, jestli během časového úseku, byl prováděn libovolný (v našem případě DoS) útok. K tomu bylo potřeba zjistit jakou velikost má komprese za stejný časový úsek v normálním provozu a následně při útoku a jejich poměr při kompresi.

Tyto dva výsledky bylo potřeba poté porovnat a zjistit, jak se výsledná komprese projevila.

Provedlo se zachytávání paketů ve dvou případech po dobu 1 minuty (vždy 5 měření). V prvním případě byl provoz síťové komunikace bez vnějšího zásahu a při druhém bylo provedeno následné zahlcení síťového provozu.

Již při běžném provozu jsme získali informace v počtu stovek tisíc jednotlivých znaků a to je již dostatečně velký počet na kompresi, při které může nastat problém. Je možné, že z tak velkého počtu znaků bude komprese trvat i několik minut.

Při provádění útoku a následného nahrání IP adres do vstupu programu, můžeme zjistit počet prvků. Podle vložených příkladů, je zde více než milion znaků (v předchozím případě bylo řádově stovky tisíc). Tento proces může trvat mnohem déle, jelikož je potřeba zjistit četnost prvků a to vyžaduje projití všech jednotlivých znaků.

Z tohoto důvodu byla upravena vyzuální stránka a některé procesy se vytvořily na zvláštní tlačítka.

Při běžném provozu se komprese pomocí Huffmanovi metody snížila přibližně o 22% ve většině případů. Pokud bychom porovnali s daty, která jsou při útoku, tak se komprese dokázala snížit o více než 27% viz kapitola 19.

Můžeme tedy říci, že komprese pomocí Huffmanovi metody dokázala snížit velikost o více procent z důvodu častého opakování znaků při zasílání DoS na koncovou stanici. Pokud by probíhal útok po delší domu, je více než pravděpodobné, že se sníží o více procent. Musí se brát také důraz na komprimovaný časový úsek a také jak dlouho probíhal DoS útok. Čím kratší bude, tím se komprese bude rovnat běžnému provozu.

Pokud bychom měli používat LZ77 metodu na kompresi dat, které obsahují IP adresy, je potřeba v lepším případě vytvořit větší prohlížecí a aktuální okna. Tento způsob nemusí být ale dobrý, kvůli rychlosti a nutnosti procházet více dat při porovnání. Výdoné je že tato metoda je jednoprůchodová a nemusí se znát přesný počet prvků v řetězci a jejich četnost.

Literatura

[1] BALÍK, PHD., et al. 2005. LZ-77. *Knihovna vizualizačních appletů pro kompresi dat* [online]. Praha: ČVUT [cit. 2016-05-25]. Dostupné z: <http://www.stringology.org/DataCompression/lz77/index_cs.html>

[2] BALÍK, PHD., et al. 2005. Statické Huffmanovo kódování. *Knihovna vizualizačních appletů pro kompresi dat* [online]. Praha: ČVUT [cit. 2016-05-25]. Dostupné z: <http://www.stringology.org/DataCompression/sh/index_cs.html>

[3] BENEŠ, Miroslav. Slovníkové metody. *Komprese dat* [online]. Ostrava: FEI VŠB-TU [cit. 2016-05-25]. Dostupné z: <http://www.cs.vsb.cz/benes/vyuka/pte/texty/komprese/ch02s04.html>

[4] BENEŠ, Miroslav. Huffmanovo kódování. *Komprese dat* [online]. Ostrava: FEI VŠB-TU [cit. 2016-05-25]. Dostupné z: <http://www.cs.vsb.cz/benes/vyuka/pte/texty/komprese/ch02s02.html>

[5] HOKSZA, David. Http://reboot.cz/ 2002. *Statické Huffmanovo kódování.* [online]. Ostrava: FEI VŠB-TU [cit. 2016-05-25]. Dostupné z: <http://reboot.cz/howto/programovani/staticke-huffmanovo-kodovani/articles.html?i

[6] HORDĚJČUK, Ing.Vojtěch. *Huffmanovo kódování.* [online]. Praha [cit. 2016-05-25]. Dostupné z: <http://voho.cz/wiki/kodovani-huffman/>

[7] HORDĚJČUK, Ing.Vojtěch. *Kompresní algoritmus LZ77.* [online]. Praha [cit. 2016-05-25]. Dostupné z: <http://voho.cz/wiki/algoritmus-lz77/>

[8] HORDĚJČUK, Ing.Vojtěch. *Kompresní algoritmus LZW.* [online]. Praha [cit. 2016-05-25]. Dostupné z: <http://voho.cz/wiki/algoritmus-lzw/>

[9] KOLOUCH, JUDr.Jan. *Kybernetické útoky.* [online]. Praha [cit. 2016-05-25]. Dostupné z: <https://csirt.cesnet.cz/_media/cs/documents/kyberneticke_utoky.pdf>

[10] NOVÁK, T. *LZ 77.* [online]. Praha [cit. 2016-05-25]. Dostupné z: <https://atrey.karlin.mff.cuni.cz/~tnovak/compress/lz77/>

[11] NOVÁK, T. *Huffmanovo kódování.* [online]. Praha [cit. 2016-05-25]. Dostupné z: <https://atrey.karlin.mff.cuni.cz/~tnovak/compress/huffman/>

[12] PETYOVSKÝ, P. *Metody a algoritmy komprese dat: Od základních principů k aplikaci.* [online]. Praha [cit. 2016-05-25]. Dostupné z: <http://zxm.speccy.cz/files/clanky/jhcon_formaty_komprese_2008sirk_rev15.pdf>

[13] SPEROTTO, A, et al. 2010. An Overview of IP Flow-Based Itrusion Detection. *IEEE Communications Surveys and Tutorials.* [online]. [cit. 2016-05-25].

[14] *Kybernetický útok.* [online]. Wikipedia [cit. 2016-05-25]. Dostupné z: <https://cs.wikipedia.org/wiki/Kybernetick%C3%BD_%C3%BAtok>

[15] CSRIT *Základní principy DoS útoku.* [online]. Praha [cit. 2016-05-25]. Dostupné z: <https://www.csirt.cz/page/2790/zakladni-principy-dos-utoku/>

[16] *Referenční model ISO/OSI.* [online]. Wikipedia [cit. 2016-05-25]. Dostupné z: <https://cs.wikipedia.org/wiki/Referen%C4%8Dn%C3%AD_model_ISO/OSI>

[17] *IT slovník.* [online]. Praha [cit. 2016-05-25]. Dostupné z: <http://it-slovnik.cz/>

[18] OVH *Co je ochrana anti-DDoS?* [online]. Wikipedia [cit. 2016-05-25]. Dostupné z: <https://www.ovh.cz/anti-ddos/princip-anti-ddos.xml>

[19] DoS *Denial-of-service Attack – DoS using hping3 with spoofed IP in Kali Linux* [online]. Wikipedia [cit. 2016-05-25]. Dostupné z: <http://www.blackmoreops.com/2015/04/21/denial-of-service-attack-dos-using-hping3

[20] LZ77 *Komprese LZ77 (Lempel-Ziv-Welch 84)* [online]. Wikipedia [cit. 2016-05-25]. Dostupné z: <https://cs.wikipedia.org/wiki/LZW>

17. Seznam použitých zkratek a symbolů

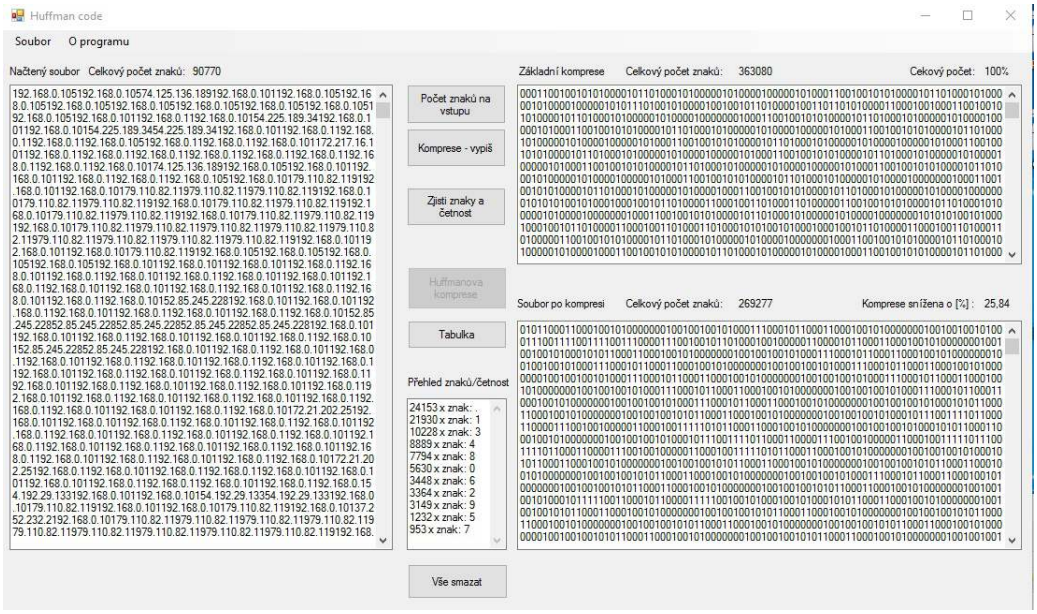
ACK	Paket který slouží k navázání komunikace
C#	Programovací jazyk
DoS	Denail of Service
DDoS	Distributed Denail of Service
flood	Parametr sloužící k zahlcení stanice
IP	Jedná se o číselnou adresu zařízení v síti
ISO-OSI	Standardizace počítačové sítě (struktura vrstev)
LZ77	Kompresní metoda podle autorů: Abraham Lemel a Jacob Ziv
LZW	Slovníková metoda
PING	Odeslání dotazu na koncové zařízení
QoS	Slouží k rezervaci a řízení datových přenosů
RFC	Popisuje dokumenty internetových protokolů
SYN	Paket sloužící k navázání komunikace
TCP	Protokol transportní vrstvy
UDP	Jeden z protokolů na komunikaci, nedává záruku na přenos
URL	V překladu se jedná o ”jednotnou adresu zdroje”

18. Seznam příloh

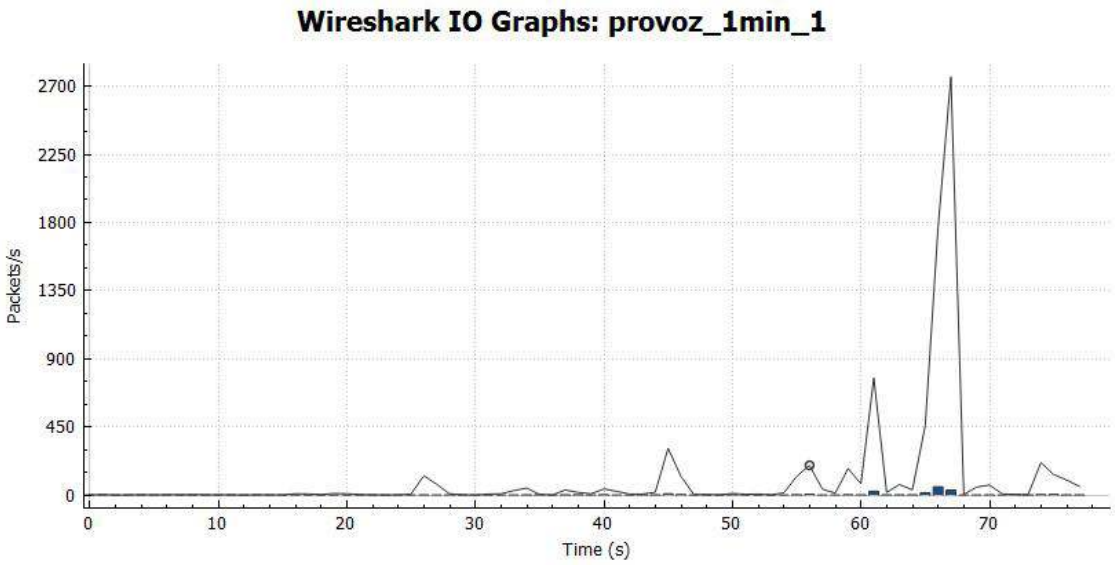
K bakalářské práci bude přiloženo CD, které bude obsahovat:

- Zdrojové kódy k programu na kompresi pomocí Huffmanovi metody a LZ 77.
- Data zachycená při komunikaci a při DoS útoku.
- Textové soubory obsahující vstupní data (IP adresy pro Huffmanovu metodu a text pro metodu LZ 77).
- Vyhotovení bakalářské práce v PDF.

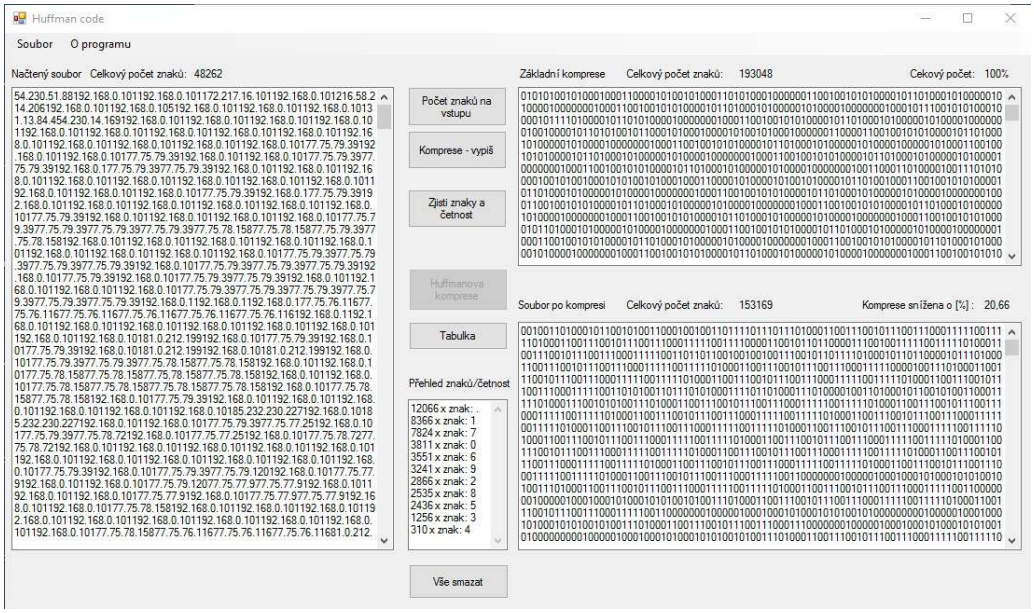
19. Příloha



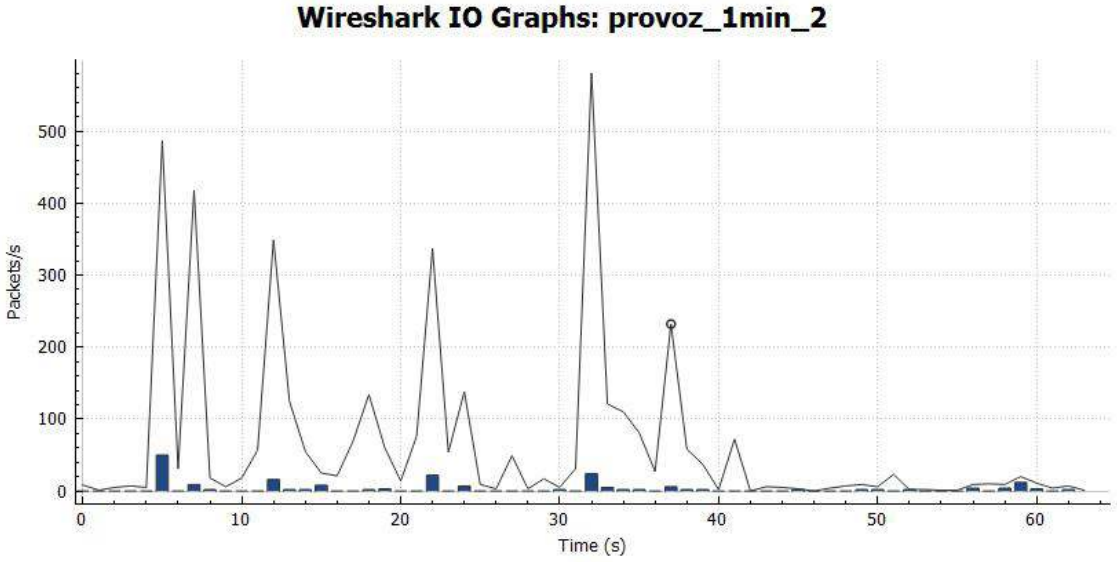
Obrázek 19.1: První měření a komprese dat.



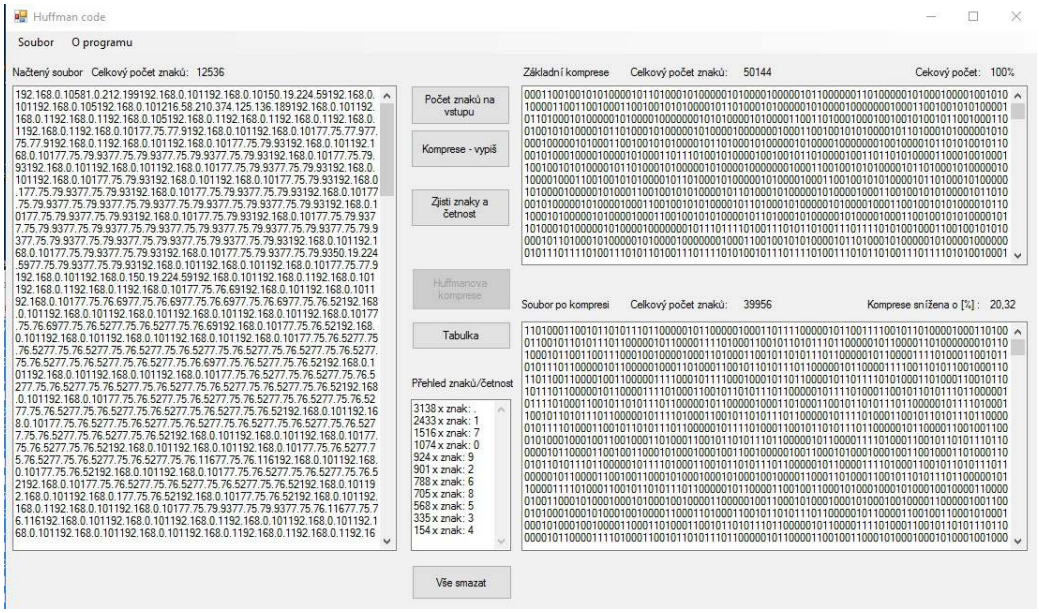
Obrázek 19.2: Graf průběhu komunikace při prvním měření.



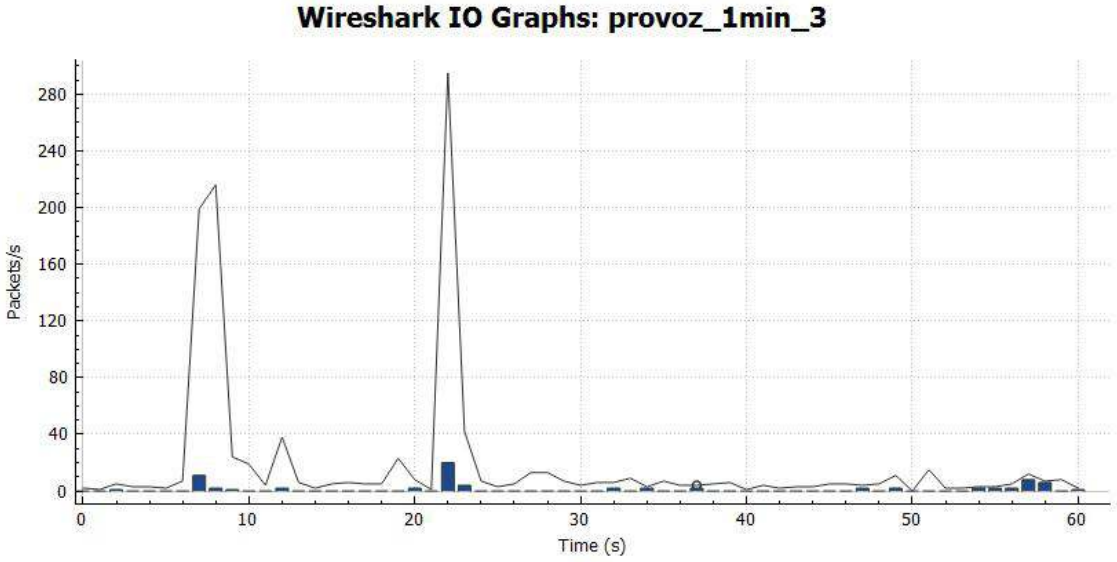
Obrázek 19.3: Druhé měření a komprese dat.



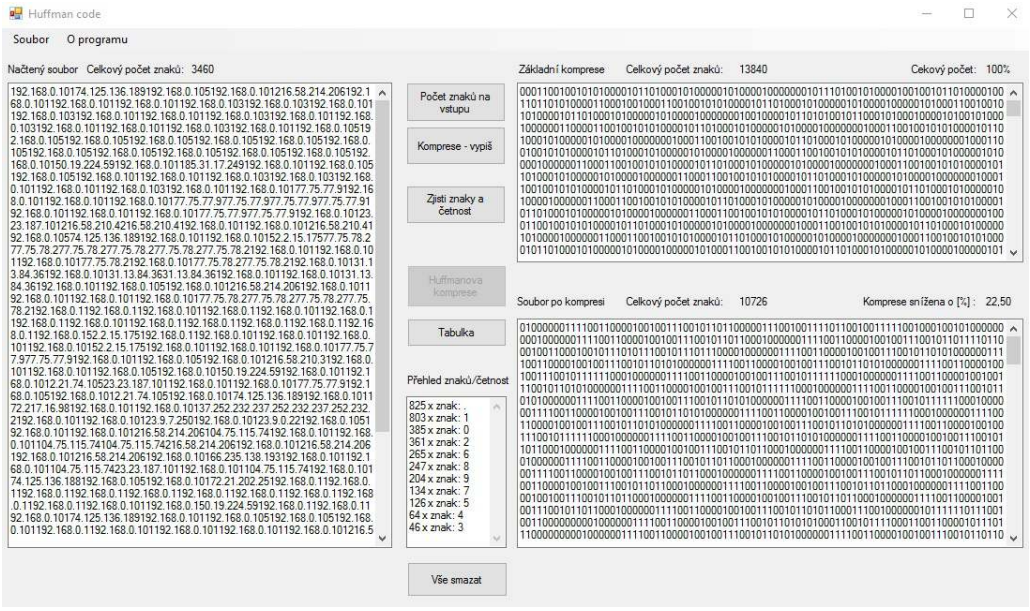
Obrázek 19.4: Graf průběhu komunikace při druhém měření.



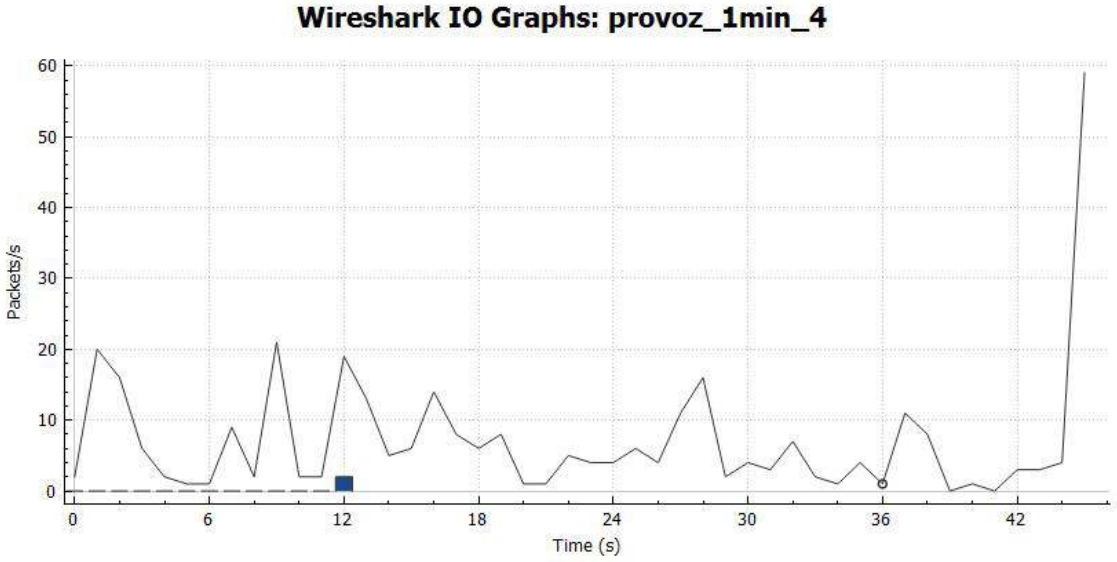
Obrázek 19.5: Třetí měření a komprese dat.



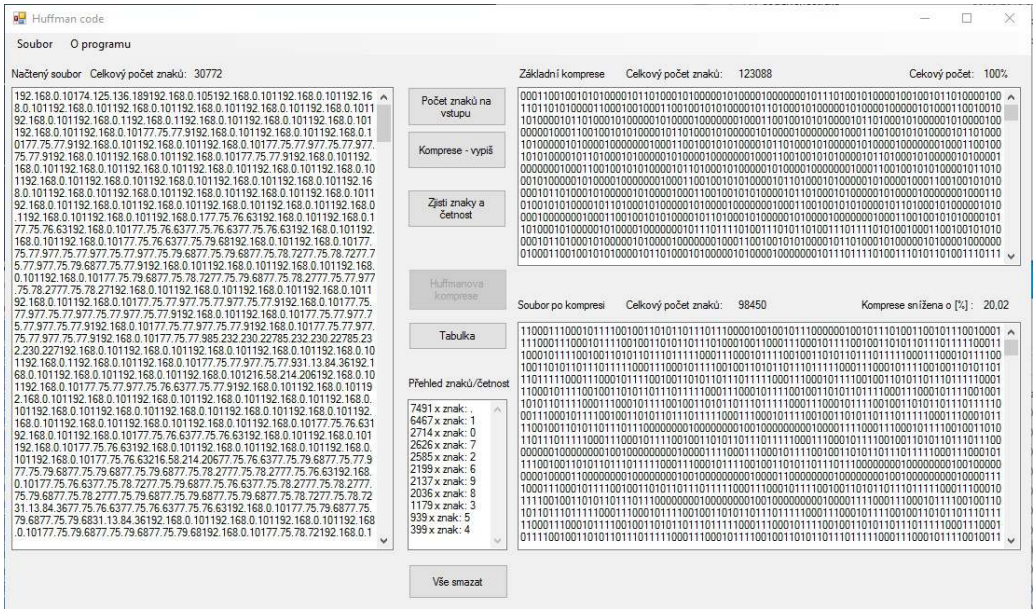
Obrázek 19.6: Graf průběhu komunikace při třetím měření.



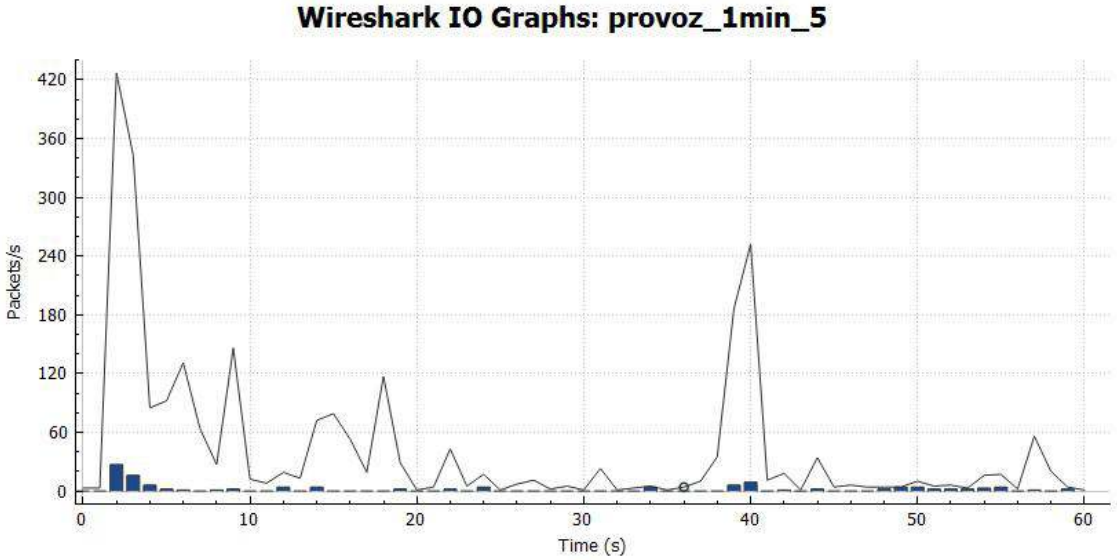
Obrázek 19.7: Čtvrté měření a komprese dat.



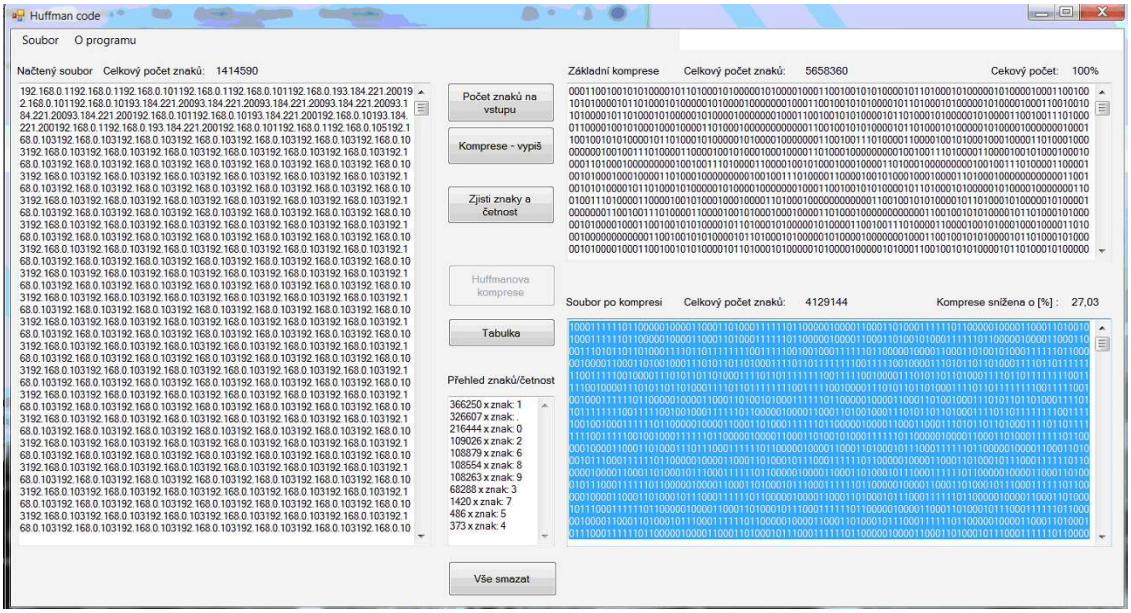
Obrázek 19.8: Graf průběhu komunikace při čtvrtém měření.



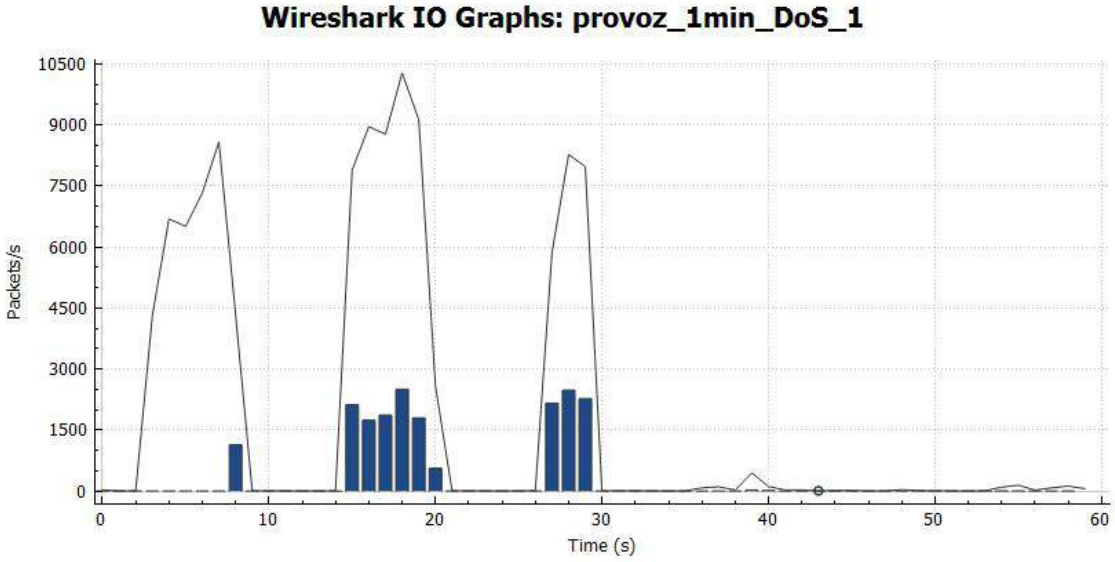
Obrázek 19.9: Páté měření a komprese dat.



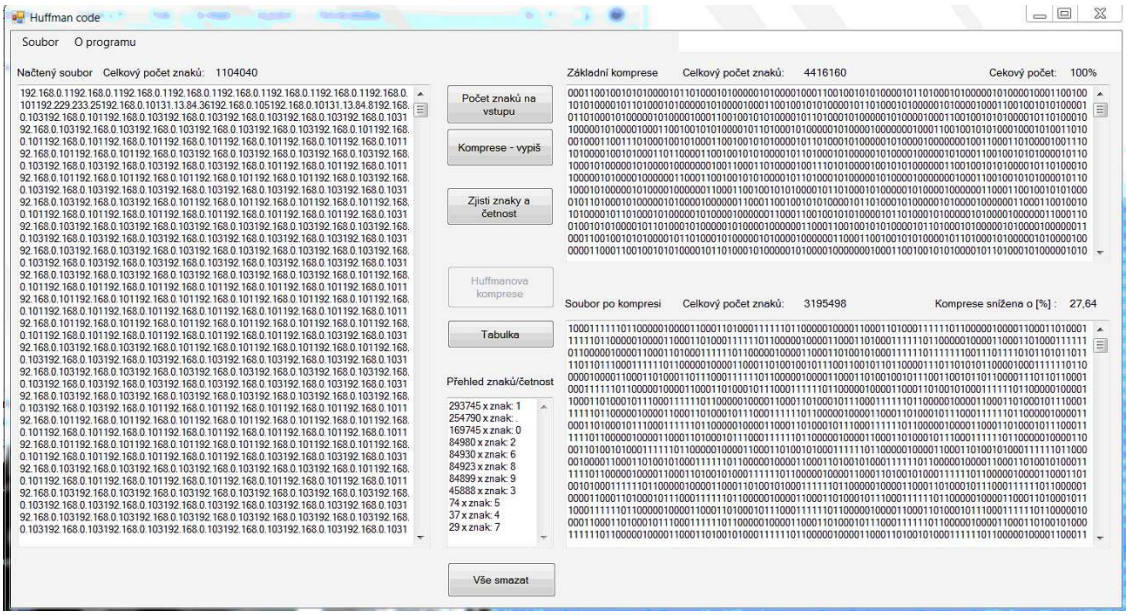
Obrázek 19.10: Graf průběhu komunikace při pátém měření.



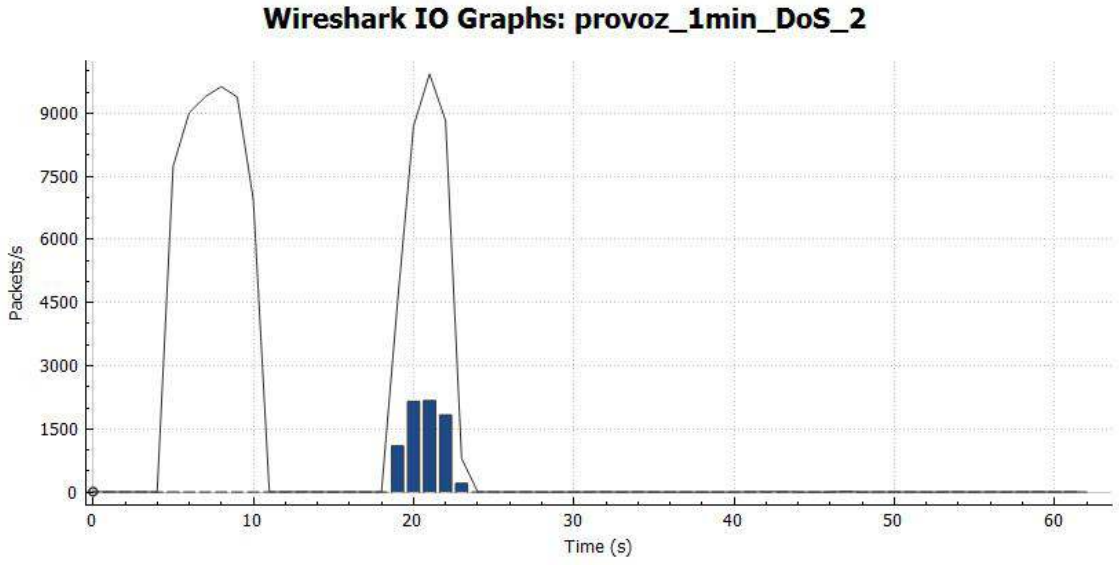
Obrázek 19.11: Komprese dat při DoS útoku.



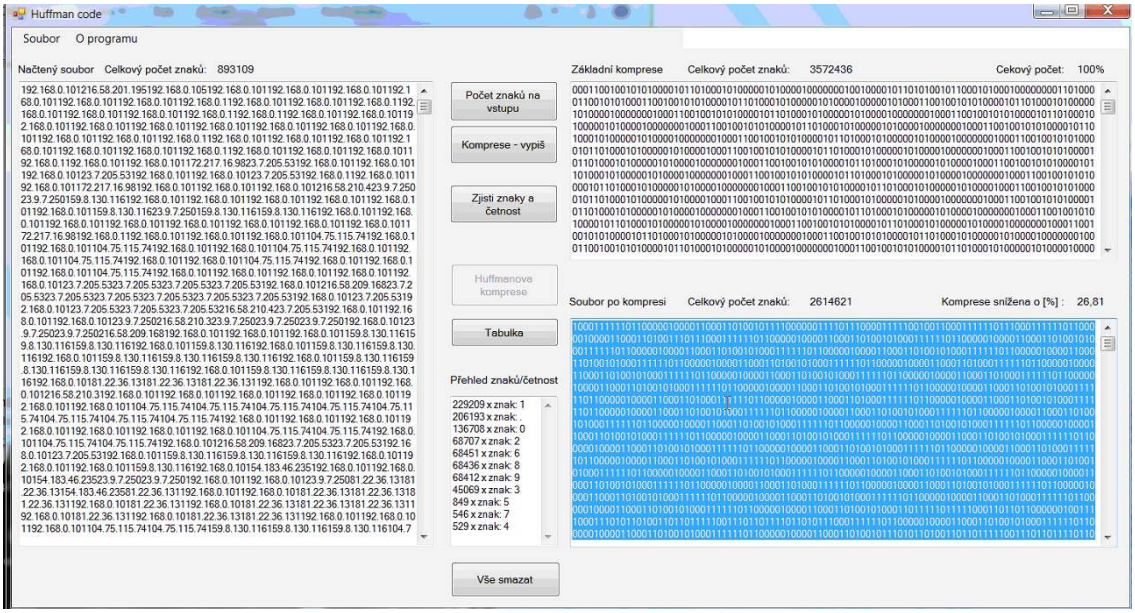
Obrázek 19.12: Průběh komunikace při třech DoS útocích.



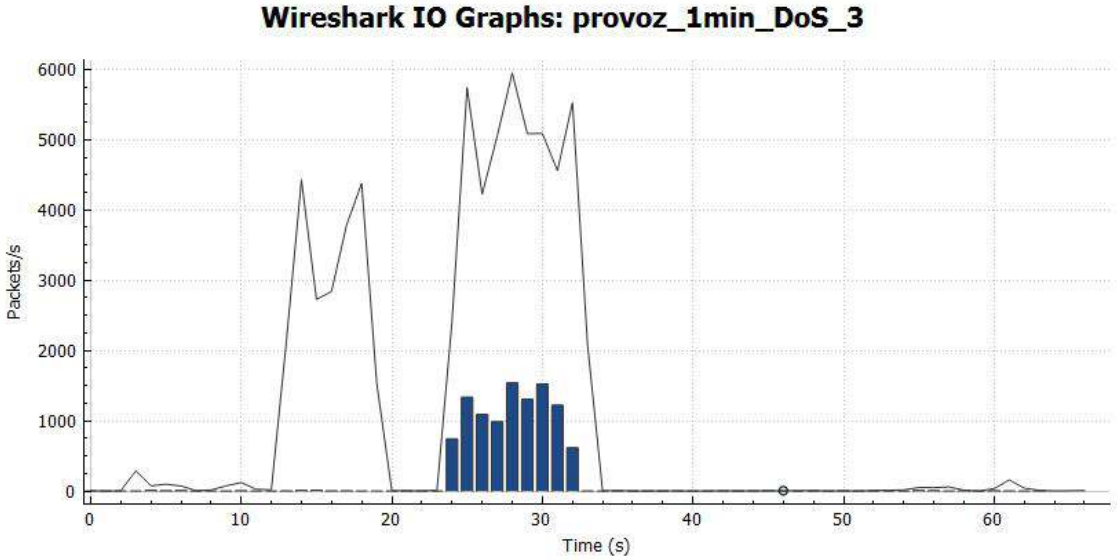
Obrázek 19.13: Komprese dat při dalším DoS útoku.



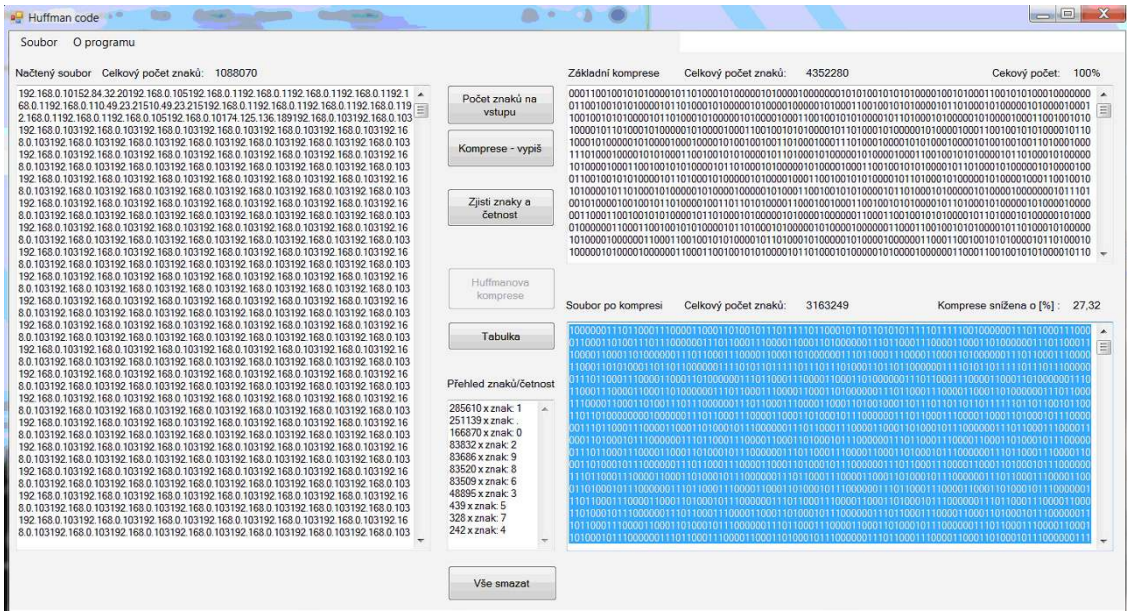
Obrázek 19.14: Průběh komunikace a dvou DoS útocích.



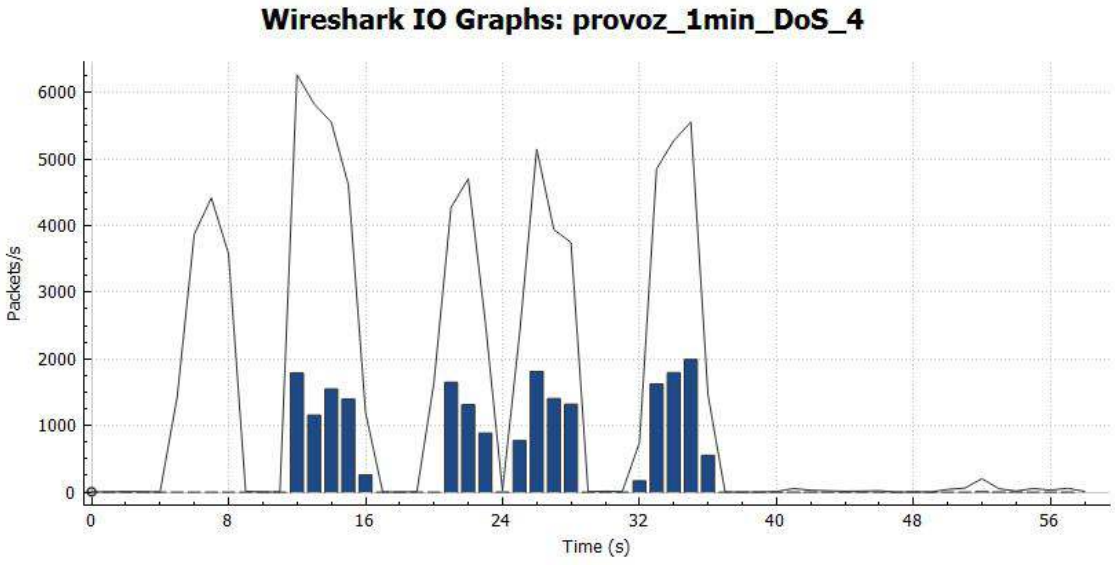
Obrázek 19.15: Komprese dat při DoS útoku podle grafu 19.16.



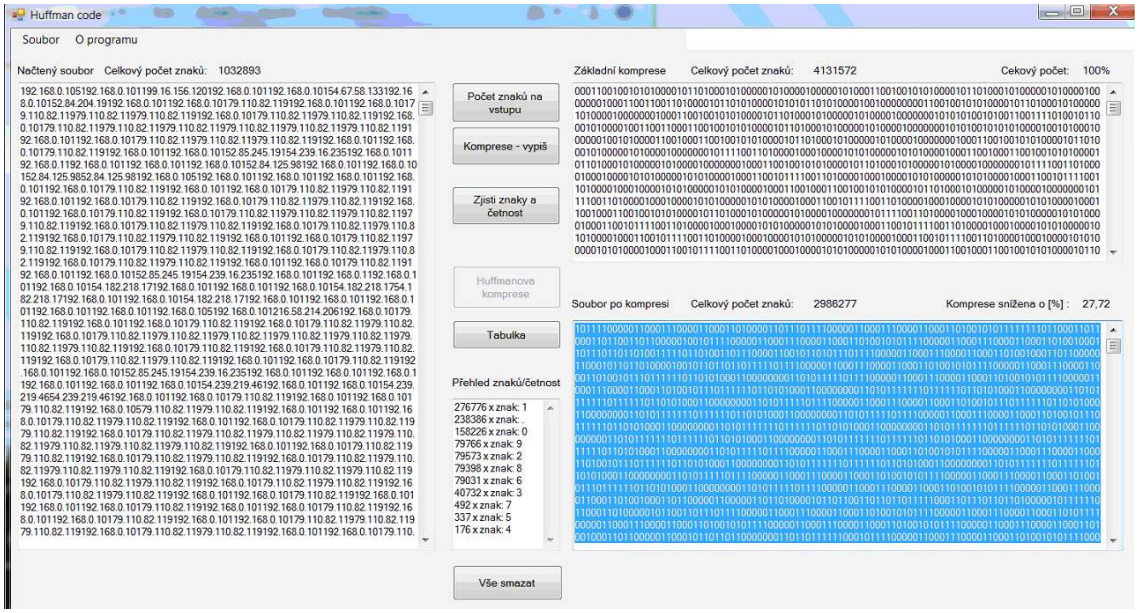
Obrázek 19.16: Průběh komunikace při dvou útocích DoS.



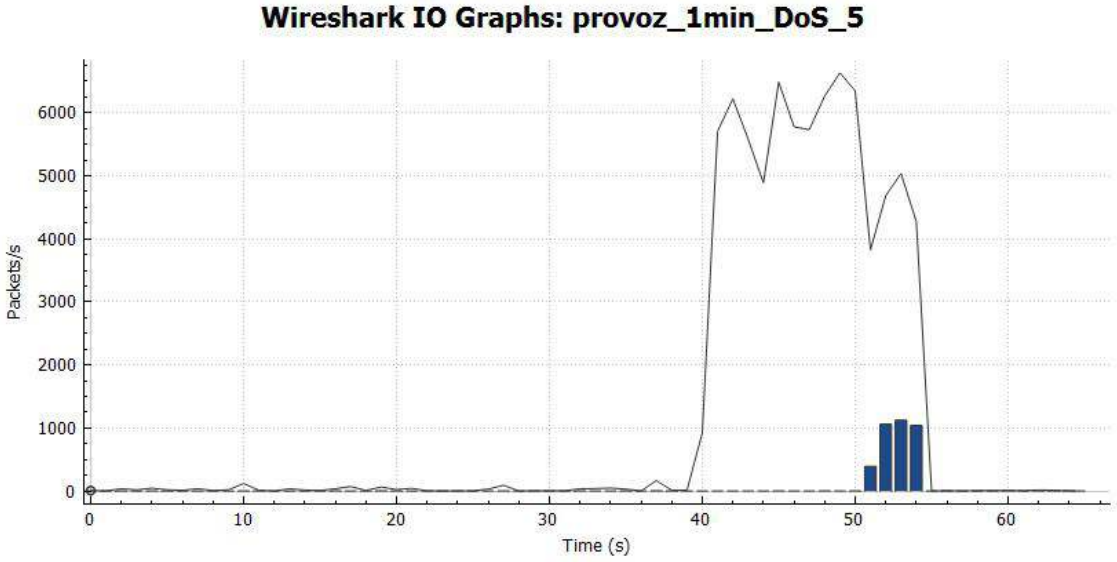
Obrázek 19.17: Komprese obsáhlejších dat při DoS.



Obrázek 19.18: Graf komunikace při DoS útoku (5 utoků). Lze vidět zahození velké části paketů.



Obrázek 19.19: Komprese dat podle zachycené komunikace viz 19.20



Obrázek 19.20: Průběh komunikace při jednom delším DoS zahlcení.