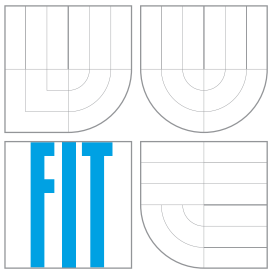
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

EXPORTOVÁNÍ OBSAHU WEBOVÉHO PROVOZU DO MAFF

WEB TRAFFIC DATA EXPORT TO MAFF

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VÍT JANEČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR VESELÝ

BRNO 2016

Abstrakt

Cílem této bakalářské práce bylo seznámení se s principy přenosu, uložení a opětovného zobrazení webového provozu. Další částí byl návrh a realizace modulu pro rekonstrukci a exportování obsahu webového provozu. Tento modul umožňuje poskládat webové stránky získané ze zachycené webové komunikace. Rovněž umožňuje takové stránky zobrazit. Výstupy modulu jsou tvořeny pomocí archivačního formátu MAFF, který slouží k uložení webové stránky pro možnost pozdějšího zobrazení. Základní ověření funkčnosti modulu probíhalo pomocí sad testů.

Abstract

The goal of this bachelor thesis was to get acquainted with the principles of transmission, storage and re-visualisation of web traffic. The next part lied with the design and realization of the module for the reconstruction and exporting of the web traffic contents. This module allows to assemble the websites acquired from the captured web communication. It also allows to visualize such websites. The outputs of the module are made through the archival format MAAF, which serves to store web pages for the option of later display. The basic validation of the module functionality was done using test sets.

Klíčová slova

Netfox Framework, Netfox Detective, archivační formáty MHTML a MAFF, HTTP, MIME, DOM

Keywords

Netfox Framework, Netfox Detective, archive formats MHTML a MAFF, HTTP, MIME, DOM

Citace

JANEČEK, Vít. *Exportování obsahu webového provozu do MAFF*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Veselý Vladimír.

Exportování obsahu webového provozu do MAFF

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod odborným vedením pana Ing. Vladimíra Veselého. Další potřebné informace mi poskytl Ing. Jan Pluskal. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vít Janeček
17. května 2016

Poděkování

Chtěl bych poděkovat mé milované rodině, která za mnou stála i v dobách psání a zpracovávání bakalářské práce. Má vděčnost patří hlavně vedoucímu práce Ing. Vladimíru Veselému a Ing. Janu Pluskalovi, který mě uvedl do bezpečnostní platformy Netfox. Další díky také patří mému kolektivu spolubydlících, kteří mě v těžkém období zásobovali rozmanitou a pestrou stravou. Touto cestou bych chtěl především poděkovat svému bratru Martinovi, který mě vždy podporoval v nejtěžších chvílích této práce naší domací kyselicou.

Valašská kyselica

- 400 g kysaného zelí a 300 g nakrájených brambor
- 3 kuličky černého pepře a 3 kuličky nového koření
- 2 hrstě sušených hub, 2 bobkové listy a 1 velká cibule
- 2 Ostravské klobásky a 100 g plátků anglické slaniny
- 300 ml mléka, 2 lžíce sádla a 1 sladká smetana (min 31%)
- 2 vrchovaté lžíce hladké mouky a 1 litr vody

Klobásu nakrájejte na půlkolečka, slaninu a cibuli nadrobno. V kastrolu rozehřejte sádlo, přihoďte cibuli, slaninu a klobásku. Restujte, zasypte moukou a za stálého míchání přelijete mléko a následně vodu. Poté přidejte koření v čajovníku a nechte vařit přibližně 45 minut. V menším kastrolu dejte vařit zelí – rozkrojte napůl, osolte, okmínujte a zalijte vroucí vodou. Brambory nakrájejte na středně velké kousky. Z polévky vylovte čajovník s kořením, přidej zelí včetně vody, brambory, smetanu a spařené houby. Vařte, dokud brambory nejsou na skus. Dochuťte a podávejte s čerstvým chlebem.
Poznámka:
Houby spaříte tak, že je osolené zalijete troškou vroucí vody a necháte odstát.

Obsah

1 Úvod **3**

2 Rekonstrukce a export webového provozu **4**
2.1 Hypertextový protokol . . . 4
2.1.1 Charakteristika a použití . . . 4
2.1.2 Základní struktura . . . 5
2.1.3 Šifrování . . . 9
2.1.4 Analýza webového provozu . . . 10
2.2 Exportní a archivační formáty . . . 11
2.2.1 Charakteristika archivačního formátu . . . 11
2.2.2 Abstraktní struktura archivačního formátu . . . 11
2.3 Archivační formát MHTML . . . 13
2.3.1 Formát uložení MIME . . . 13
2.3.2 Struktura MHTML formátu . . . 14
2.4 Mozilla Archive File Format . . . 16
2.4.1 Charakteristika formátu . . . 16
2.4.2 Základní struktura formátu . . . 16
2.4.3 Metody uložení v rámci formátu . . . 19

3 Návrh a implementace modulu pro rekonstrukci webu **20**
3.1 Analýza požadavku na návrh řešení modulu . . . 20
3.1.1 Nástroj pro analýzu Netfox Detective . . . 20
3.1.2 Rozšiřující moduly a modul pro exportování webových objektu . . . 21
3.1.3 Technologické požadavky na MAFF modul . . . 21
3.2 Návrh řešení implementace . . . 22
3.2.1 Popis implementace . . . 23
3.2.2 Zpracování vstupních objektů . . . 24
3.2.3 Typy MAFF objektů . . . 25
3.2.4 Skládání MAFF archivu . . . 26
3.2.5 komplementace a komprimace archivu pro výstup . . . 29
3.2.6 Vizualizace výstupních archivů . . . 31

4 Testování a ověření výsledků **33**
4.1 Charakteristika testů . . . 33
4.2 Výsledky testů . . . 34

5 Závěr **35**

Literatura **37**

Přílohy **39**
Seznam příloh 40

A Obsah CD **41**

Kapitola 1

Úvod

Fenoménem dnešní doby je nepřeberné množství různých moderních technologií, které ovlivňují náš svět. Jednou z největších technologií, která se dennodenně dotýká každého z nás, je Internet. Systém sloužící převážně pro okamžitou komunikaci, sdílení dat či informací v rámci celého světa, Internet často operuje s komunikacemi citlivého typu. Proto je potřeba klást velký důraz na bezpečnost každé komunikace v rámci Internetu.

Takové zabezpečené komunikace jsou bohužel často cílem různých útočníků, kteří se chtějí dopátrat obsahové části komunikace, a dokonce obsah i modifikovat. K odhalení těchto bezpečnostních hrozeb a incidentů vytvořila Fakulta informačních technologií Vysokého učení technického v Brně bezpečnostní nástroj, která k zjištění těchto hrozeb využívá forenzní analýzu. Díky tohoto bezpečnostního nástroje lze analyzovat jednotlivé aplikační protokoly a určit, zdali daná komunikace není bezpečnostní hrozbou.

Poslední dobou se útočnici často zaměřují na vytváření podvodných stránek imitující stránky původní[1]. Nejčastějším typem podvržené webové stránky je internetové bankovnictví. V případě, že se oběť přihlásí skrze podvrženou webovou stránku internetového bankovnictví, prozradí útočníkovi důvěrné informace, které lze zneužít k přístupu k účtu.

Cílem této bakalářské práce je implementace modulu pro bezpečnostní nástroj, který dokáže ze zachycené komunikace zpětně zrekonstruovat a vizualizovat jednotlivé navštívené stránky. Díky této charakteristice lze odhalit podvodné stránky, interpretovat jejich obsah a vizualizovat je. Vizualizace a interpretace obsahu dokážou být nápomocny k zjištění identity útočníka a slouží také jako preventivní ochrana dalších bezpečnostních hrozeb.

Samotná bakalářská práce pak popisuje prerekvizity a principy potřebné pro návrh a realizaci modulu a jeho implementaci části v bezpečnostním nástroji. Prerekvizity pro návrh modulu jsou uvedené v kapitole 2, která popisuje přenos webového provozu prostřednictvím protokolu HTTP, jeho možnost zabezpečení a formáty pro uložení obsahu těchto webových provozů. Návrh a realizace modulu je nastíněna v kapitole 3. Kapitola nejprve popisuje implementační nároky na technologie potřebné k vytvoření a vizualizaci zachycených webových stránek. Poté se zabývá implementační části modulu, kde se zaměřuje na principy sestavení webových stránek do cílové podoby. Další část kapitoly popisuje principy uložení jednotlivé webové stránky skrze archivační formát MAFF, který tvoří výstupy implementovaného modulu. V poslední části kapitoly jsou popsány principy testování a ověření funkčnosti modulu.

[1] *Phising* - http://www.phishing.org/phishing-techniques/

Kapitola 2

Rekonstrukce a export webového provozu

Hlavní náplní této kapitoly jsou základní informace o přenosu dat pro vykreslení a načtení kterékoliv webové stránky či aplikace. Kapitola se zaměří na strukturu protokolu, který je využíván pro přenos jakéhokoliv souboru či formátu pro získání obsahu webové stránky. Dále uvede nástroje, které tento protokol dokáží analyzovat. Především pak přinese čtenáři základní podvědomí o fungování přenosu webových dat, možnostech jejich interpretace a možnostech jejich uložení a výsledného zobrazení.

2.1 Hypertextový protokol

Hypertext Transfer Protocol, dále jen HTTP[11], je protokol na úrovni aplikační vrstvy v ISO/OSI[1]modelu. Protokol je sestaven pro přenos distribuovaných, multimediálních dat a informací. Je generickým, textově orientovaným, bezstavovým protokolem, který je využíván pro přenos hypertextových dokumentů, a to pomocí jeho základního rozdělení na dotaz či odpověď popsaný v oddílu 2.1.2.

2.1.1 Charakteristika a použití

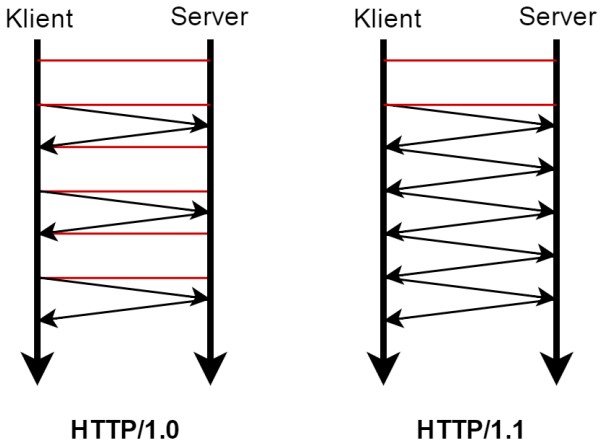
Charakteristika HTTP udává, že protokol je používán pro přenos multimediálních a hypertextových dat, které se využívají při zobrazení webové stránky. Při zobrazení webové stránky jsou přenášené hypertextové dokumenty jako například *Hypertext Markup Language*, dále jen HTML[2], kaskádové styly, dále jen CSS, a různé skriptovací enginy, jako Javascript, JQuerry, AJAX a další. Protokol také dokáže přenášet libovolný typ multimediálních dat, a to pomocí rozšíření MIME, které si více přiblížíme v oddílu 2.3.1.

HTTP protokol využívá komunikačního modelu klient-server. Charakteristikou této architektury je, že strana klienta zasílá požadavky serveru a čeká na jejich odpověď. Server požadavek zpracuje, vyhodnotí a pošle požadovanou odpověď zpět. Každý dotaz a odpověď společně tvoří tzv. konverzaci. Každá tato konverzace je nezávislá na předchozí a díky tomu je HTTP bezstavovým protokolem. Odpověď serveru typicky interpretuje, jestli se požadavek splnil a obnáší požadovaná data. Splnění požadavku se kontroluje skrze chybové kódy, tzv. *error codes*, které jsou obsaženy v každé odpovědi.

[1]ISO/OSI - https://www.ietf.org/rfc/rfc1208.txt
[2]HTML- https://tools.ietf.org/html/rfc1866

Klient při navázání komunikace se serverem vytvoří tzv. spojení, ve kterém se přenáší požadavky a odpovědi. Pomocí staršího standardu HTTP/1.0 pak bylo pro každý požadavek a odpověď (konverzace) vytvořeno nové spojení [6]. Po dokončení odpovědi se toto spojení uzavřelo. Tento model měl díky svému návrhu obrovskou režii, která zbytečně zatěžovala obě strany při opětovném vytváření nových spojení. Proto se zavedl novější standard HTTP/1.1, který má každé spojení perzistentní [11]. To znamená, že libovolný počet požadavků a odpovědí může být přenesen skrze jediné spojení. Spojení se ukončí až v případě, kdy si to vyžádá klient nebo server.

Rozdíl mezi verzemi standardu je zachycen na obrázku 2.1, kde lze u první verze protokolu vidět po přenosu jednoho objektu režii kvůli opětovnému vytvoření nového spojení. Oproti tomu jde u verze HTTP/1.1 vidět režii pouze při vytváření spojení a všechny objekty jsou přenesené jediným spojením.



Obrázek 2.1: Rozdíl mezi HTTP/1.0 a HTTP/1.1

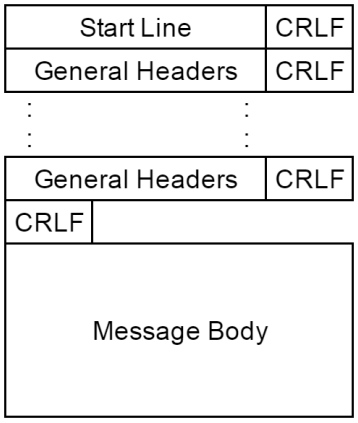
Použití HTTP protokolu v dnešní době je enormní. Nejenže slouží jako hlavní opěrný bod komunikace na Internetu, ale jak již bylo zmíněno, je využíván při přenosu webových stránek či dat ve webových aplikacích. Také slouží pro stahování či nahrávání souborů na server. Rovněž umožňuje přenášet dynamické skripty[3], které upravují výsledné zobrazení stránky přímo u klienta a také šetří výkonnost serveru.

2.1.2 Základní struktura

HTTP protokol přenáší data pomocí dvojic zpráv dotaz (request) a odpověď (response). Požadavek i odpověď mají stejnou základní strukturu, které se říká HTTP zpráva. Každá zpráva se dělí na hlavičku zprávy a tělo zprávy. Hlavička přenáší pevně definované informace, které se dělí dle toho, zdali se jedná o hlavičku požadavku či odpovědi, a ukončuje se textovou sekvencí CRLF[4]na samostatném řádku. Poté následuje tělo zprávy, které obsahuje data specifikované v hlavičce. HTTP zpráva je znázorněna na obrázku 2.2.

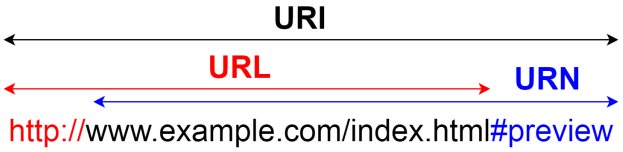
[3]Skripty obsahují zdrojový kód programu, který umožnuje dynamicky měnit data a podobu prohlížené stránky

[4]Textová sekvence CRLF znamená odřadkování neboli nový řádek a je zakodována pomocí znaků Carriage Return (ASCII kód 0x0D) a Line Feed (ASCII kód 0x0A)



Obrázek 2.2: Struktura HTTP zprávy

Skladba HTTP požadavku je podobná základní skladbě zprávy. Každý požadavek má tzv. *request line*, která obsahuje metodu požadavku, umístění zdroje a verzi protokolu. Umístění zdroje je definováno pomocí *Universal Resource Identifier*, dále jen URI, tedy sekvence znaku, které jsou použité pro identifikaci jména či zdroje na Internetu. URI se dělí na *Uniform Resource Locator*, dále jen URL, a *Uniform Resorce Name*, dále jen URN. URL je podmnožinou URI a specifikuje, kde je zdroj dostupný, zatímco URN specifikuje identifikaci zdroje. URL a URN dohromady tvoří URI [4]. Na obrázku 2.3 lze vidět zobrazení URI a jejich částí.



Obrázek 2.3: Ukázka rozdělení URI

Dále se v *request line* nalézá metoda požadavku, která se rozděluje do následujících typů. Tyto typy nejsou pro zachování jejich podstaty přeloženy[11].

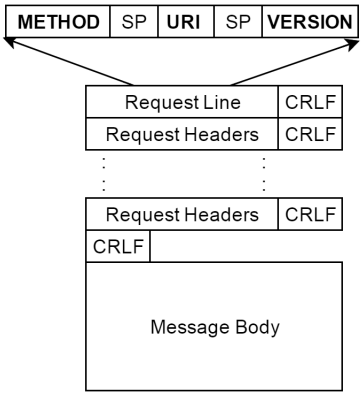
- POST metoda se využívá k odesílání informací či hypertextových dokumentů. Nejčastějším využitím je odesílání dat a informaci z hypertextového formuláře na server. Také se používá k posílání a dotazování dalších informací potřebných pro skripty, tak aby správně zobrazovaly požadované uživatelské akce.
- GET je základní metodou stahující požadované hypertextové zdroje skrze specifikovanou URI. Pomoci URI se specifikuje cílový server a hypertextový dokument, který je třeba získat.
- HEAD je identická k metodě GET, s tím rozdílem, že nevrací požadovaný hypertextový dokument, ale pouze jen hlavičku. Používá se při testování dostupnosti požadovaných zdrojů.
- OPTION metoda představuje žádost o poskytnutí informací o možnostech komunikace, které jsou k dispozici. Pomocí URI se specifikuje zdroj, kterému bude poslán dotaz o podporované metody cílového serveru.

- CONNECT metoda se používá pro připojení k Proxy, díky které lze spojení tunelovat.
- PUT metoda pomocí URI se vytvoří a nahraje dokument, na server. Používá se zřídka.
- DELETE metoda pomocí URI specifikuje, který dokument má být smazán.
- TRACE metoda se používá k zjištění cíle požadavku na aplikační vrstvě serveru. Umožní klientovi vidět, co je přijato na druhé straně kanálu, využívá se k testování.

Požadavek dále obsahuje rozšiřující pole jeho hlaviček. Tyto hlavičky obsahují doplňující informace jednotlivého požadavku. Pro danou problematiku jsou důležité následující[11]:

- *Host* specifikuje internetový zdroj, ze kterého klient požaduje daná data. Zdroj je specifikovaný pomocí URI, který se poté překládá na IP adresu a port.
- *Cookie* hlavička obsahující informace, které se nazývají *cookies* a slouží k zapamatování těch předchozích akcí, které mohou ovlivnit akce následující. Informace je nutno zapamatovat, poněvadž HTTP protokol je bezstavový. *Cookies* často uvádějící, zdali je uživatel již přihlášen či nikoliv a nastaví se pomocí *Set-Cookie* hlavičky v odpovědi na některé z předchozích požadavků.
- *Referrer* popisuje pomocí URL, ze kterého předchozího zdroje je požadován zdroj následující. V praxi to znamená, že v případě přechodu z jedné webové stránky na druhou se tato informace posílá ze stránky přístupu. Tak lze například zjistit, ze které stránky uživatel přistoupil či z jakého zdroje pochází následující požadavek.

Na obrázku 2.4 je zobrazena základní abstraktní struktura požadavku. Požadavek vychází z HTTP zprávy, ale některá pole jsou určené přímo pro požadavek. U požadavku už není prvním řádkem *status line*, ale *request line*. *Request line* je vyňatý a rozdělený na jednotlivé části. Oddělovač mezera *space*, dále jen SP, navíc odděluje každou část.



Obrázek 2.4: Struktura HTTP požadavku

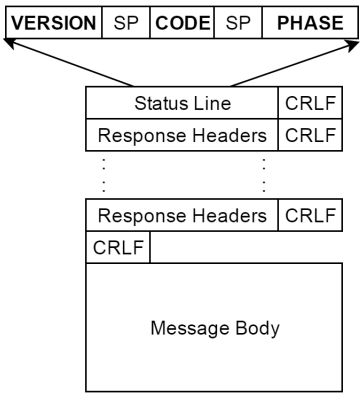
Druhou částí konverzace tvoří HTTP odpověď. Struktura požadavku také vychází ze základní struktury HTTP zprávy, kterou upravuje na požadovanou formu odpovědi. Základní část požadavku tvoří tzv. *status line*, dělen na verzi protokolu, dále tzv. *status code*, který může být chápan jako návratový kód požadavku a nakonec název návratového kódu, který se uvádí názvem *Phase* [11].

Mezi nejčastější návratové kódy patří [11]:

- Návratový kód 200 – Jeho textová interpretace *OK* znamená, že požadovaný požadavek byl splněn a odpověď lze považovat za úspěšnou.
- Návratový kód 201 – Jeho textová interpretace *Created* znamená, že požadované vytvoření hypertextového objektu bylo úspěšné.
- Návratový kód 301 – Textová interpretace kódu je *Moved Permanently*. Popisuje, že daný zdroj je přesměrován a jako odpověď v těle často udává novou URI zdroje.
- Návratový kód 304 – Textová interpretace kódu je *Not Modified*. Většinou nastává při znovunačtení takového požadovaného zdroje, který se od posledního načtení nezměnil. Do značné míry je ovlivněn *cookies* informacemi.
- Návratový kód 404 – Textová interpretace kódu je *Not Found* a vyskytuje se v případě, že požadovaný zdroj neexistuje.

Další části odpovědi jsou její hlavičky, nesoucí rozšířené informaci o odpovědi. Tyto informace můžou složit také k lepšímu popsání a definici požadovaných dat, nalezitelných jak v případě úspěšné, tak neúspěšné odpovědi. Nejčastější hlavičky odpovědi [11]:

- *Set-Cookie* – nastavuje *cookies* informace, které slouží k zapamatování informací. Ty se poté posílají v hlavičce požadavku Cookie popsané výše.
- *Content-Type* – určuje typ hypertextového dokumentu či dat, které jsou prostřednictvím odpovědí zaslány. Tato hlavička slouží k rozpoznání potřebného typu dokumentu.
- *Content-Lenght* – určuje celkovou délku požadovaného dokumentu v bytech.
- *Content-Encoding* – v případě, že navrácený objekt je textového typu, může mít svou specifickou znakovou sadu potřebnou pro správné zobrazení dat.
- *Last-Modified* – Informace, kdy byl požadovaný objekt naposledy změněn. Má široké využití při znovunačtení požadovaných dat v případě generovaní pomocí skriptovacího enginu.



Obrázek 2.5: Struktura HTTP odpovědi

Celkovou strukturu odpovědi, lze vidět na obrázku 2.5. Tato struktura vychází ze základní HTTP zprávy, ale není zde již obsažena *start line*, pouze popsaný *status line*. *Status line* je zde opět vyjmutý a dříve uvedené části popsány výše.

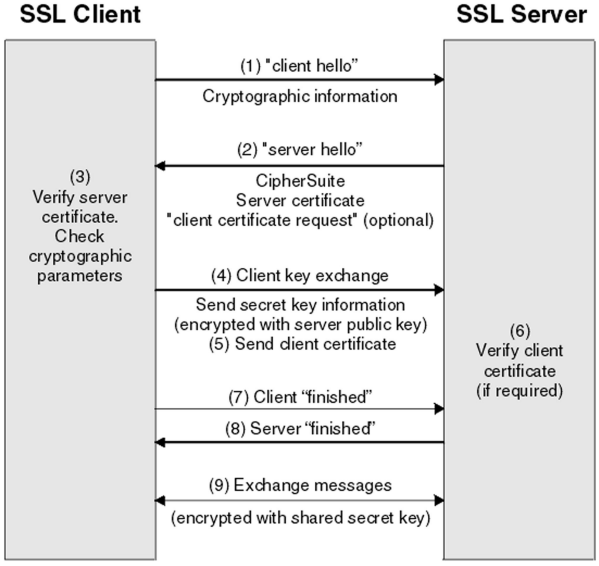
2.1.3 Šifrování

Hypertextové dokumenty a multimediální data je potřeba zabezpečit při přenosu tak, aby nemohly být odcizeny a zneužity. HTTP protokol nemá implicitní ochranu dat, a proto byl zaveden *Hypertext Transfer Protokol over TLS*, dále jen HTTPS [14]. HTTPS prostřednictvím TLS protokolu vytvoří zabezpečené spojení přes které je přenášen HTTP.

Šifrování je zajištěno pomocí *Transport Layer Security* dále jen TLS protokolu. Tento protokol leží na úrovni relační vrstvy ISO/OSI modelu. TLS protokol poskytuje zabezpečenou komunikaci skrze Internet. Protokol umožnuje aplikacím client/server, aby komunikoval způsobem, který zabraní odposlechu, padělaní a krádeži informací [10]. Nejnovější specifikace verze je TLS 1.2. Technologie TLS je novější varianta *Secure Socket Layer*, dále SSL[5], která zajištuje šifrované spojení pomocí mezivrstvy mezi transportní a aplikační vrstvou. V dnešní době je technologie SSL zastaralá a používá se TLS technologie.

Základní schéma TLS spočívá v tom, že vytváří zabezpečenou komunikaci mezi koncovými body, kde každá komunikace je zašifrovaná pomocí sdíleného privatního klíče. Každý sdílený klíč je jedinečný pro každou komunikaci a je vyjednán pomocí *TLS Handshake*[10]. *TLS Handshake* umožňuje klientovi i serveru ověřit se navzájem a dohodnout si kryptografické klíče, šifrovací sady a algoritmy ještě předtím, než se pošlou jakákoliv data. K tomu využívají mechanismus dvou asymetrických klíčů pro klienta i server. Jeden se nazývá privátní klíč, druhý pak klíč veřejný. Na počátku vytváření nového zabezpečeného spojení si klient a server dohodne vzájemné šifrovací sady a klient získá certifikát serveru, kde je uložen veřejný klíč serveru. Certifikát si klient ověří, vytvoří základ sdíleného klíče pro šifrování komunikace a ten pošle serveru jako zprávu, která je zašifrovaná pomocí veřejného klíče serveru. Zprávu lze rozšifrovat pouze privátním klíčem serveru. Díky tomu server přečte základ sdíleného klíče, doplní ho a pošle ověření, že komunikace bude šifrována tímto vytvořeným sdíleným klíčem.

Proces vyjednávání šifrovaného spojení lze vidět na obrázku 2.6. Jednotlivé kroky, které jsou zde zobrazeny, najdete detailně popsané v RFC 5246[10].



Obrázek 2.6: Ukázka SSL/TLS handshake[2]

[5]SSL - https://tools.ietf.org/html/rfc6101

2.1.4 Analýza webového provozu

Analýzou se rozumí zkoumání získaných informací v podobě zachycených dat, událostí či toků. Tyto informace lze poté interpretovat pomoci různých procesů, které získají požadovaný výsledek analýzy. Analýza webového provozu se zabývá zkoumáním hypertextových dokumentů a jejich obsahu pomocí procesů přibližujících požadovanou interpretaci dat. Získávání požadovaných informací ze sítového provozu se rozděluje na dva základní přístupy. První z nich je analýza aplikačního toku, která umožňuje identifikovat způsob zpracování informací a dat se zaměřením na jejich přenos, výměnu a uchovávání. To znamená, že aplikační tok přestavuje abstraktní celek komunikace, kde není potřeba znát nižší části komunikace, dostačující jsou totiž informace o nejvyšší (aplikační) vrstvě. Například při analýze toku HTTP protokolu není potřeba znát nižší implementační detaily, jako jsou adresa IP či port, ale postačují data, která přímo souvisí s protokolem, například dotaz či odpověď protokolu. K analýze aplikačního toku je potřeba mít sondu či analyzátor nad transportní vrstvou ISO/OSI modelu.

Druhý přístup analýzy sítového provozu se nazývá zachytávání paketu. Tento přístup je rozšířenější, protože ve stejné podobě se posílají data skrze Internet. Výhodou je plný rozsah komunikace od nejnižších po nejvyšší vrstvy komunikace. Velmi dobře lze zkoumat různé části podle daných vrstev, kdy lze použít všechny dostupné informace pro interpretaci co nejkvalitnějších výsledků. Problém této analýzy je náročnost na výpočetní zdroje, kdy například zjišťování nebezpečných komunikací v rámci síťového uzlu může být značně náročné.

Nejznámější nástroj pro analýzu sítového toku je Wireshark. Wireshark je protokolový analyzátor, který slouží pro zachycení paketu a jejich detailní zobrazení [19]. Zobrazuje veškerou komunikaci na vybraném síťovém rozhraní. Pro analýzu vybraného protokolu slouží filtry. Filtr definuje, které protokoly lze snímat či jaké části daného protokolu lze zobrazit. Pro zobrazení webového provozu v zachycené síťové komunikaci se nastavuje hodnota filtru *http*. V případě analýzy požadavků komunikace se filtr nastaví na hodnotu *http.request*.

Na obrázku 2.7 je zachycen požadavek v podobě struktury popsané v oddílu 2.1.2. Lze vidět, že druhá část konverzace, tzv. odpověď, leží v paketu 188. Požadavek obsahuje metodu GET, která obsahuje identifikaci objektu pomocí definovaného URL, a to ze zdroje specifikovaného hlavičkou *host*. Tento požadavek slouží pro stažení kořenového hypertextového dokumentu ze specifikovaného zdroje, v tomto případě www.example.com.

```
Hypertext Transfer Protocol
▹ GET / HTTP/1.1\r\n
  Host: www.example.com\r\n
  Connection: keep-alive\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome\r\n
  Accept-Encoding: gzip, deflate, lzma, sdch\r\n
  Accept-Language: en-US,en;q=0.8\r\n
  \r\n
  [Full request URI: http://www.example.com/]
  [HTTP request 1/2]
  [Response in frame: 188]
  [Next request in frame: 190]
```

Obrázek 2.7: Ukázka zachyceného požadavku

2.2 Exportní a archivační formáty

Export, též znám jako archivace, slouží pro uložení obsahu webového provozu. Ačkoliv se zdají tyto dva termíny shodné, pro lepší pochopení jejich požadavků je zavedeno dvojí názvosloví. První termín je tzv. *exportní formát* a ten slouží k uložení výsledných dat pro pozdější použití. Exportní formát je kolekce informací, dat či souborů, které jsou uložené pro analýzu, interpretaci či zobrazení datové sémantiky. Ve vztahu k dané problematice je výstupem exportní formát, který má v sobě uložen webový provoz.

V oddílu 2.1.4 byl popsán nástroj Wireshark, který dokáže uložit zachycenou síťovou komunikaci do exportního formátu PCAP [20]. Exportních formátů je vícero druhů, a není třeba je všechny rozvádět a popisovat, protože jejich základní struktura je ve své podstatě stejná. Data jsou uložena ve formě posloupnosti rámců, seřazené podle zachycení v daném čase. Každý exportní formát je perzistentní. To znamená, že uložená data mohou být kdykoliv znovu analyzována a jejich sémantická struktura znovu zobrazena.

2.2.1 Charakteristika archivačního formátu

Druhým termínem, specifikovaný pomocí jeho charakteristických vlastností je *archivační formát*. Tento formát je stejně jako *exportní formát* tvořen kolekcí dat či souborů, ale jeho primární vlastností je kromě trvalého uložení dat jejich vizualizace. Na rozdíl od *exportního formátu*, kde se vizualizuje pouze sémantika dat, dokáže *archivační formát* zobrazit původní vizualizaci tak, jak ji viděl uživatel. Tato vlastnost je do jisté míry abstraktní, protože plné zobrazení původní vizualizace je téměř nedosažitelné. Důležité je najít bod, kdy jsou požadavky na vizualizaci splněny stejně jako požadavky na časovou náročnost.

V případě, že uživatel navštívil webovou stránku www.example.com, bude v *exportním formátu* uložena posloupnost paketů, díky které lze zobrazit, jaké soubory a data se přenesly od serveru ke klientovi. Oproti tomu jsou v *archivačním formátu* tyto informace implicitně skryté, při otevření se však opět zobrazí načtená webová stránka tak, jak ji uživatel navštívil. Vzhledem k problematice zpětné rekonstrukce a vizualizace se dále bude využívat *archivační formát*, který bude nezbytnou součástí exportního modulu.

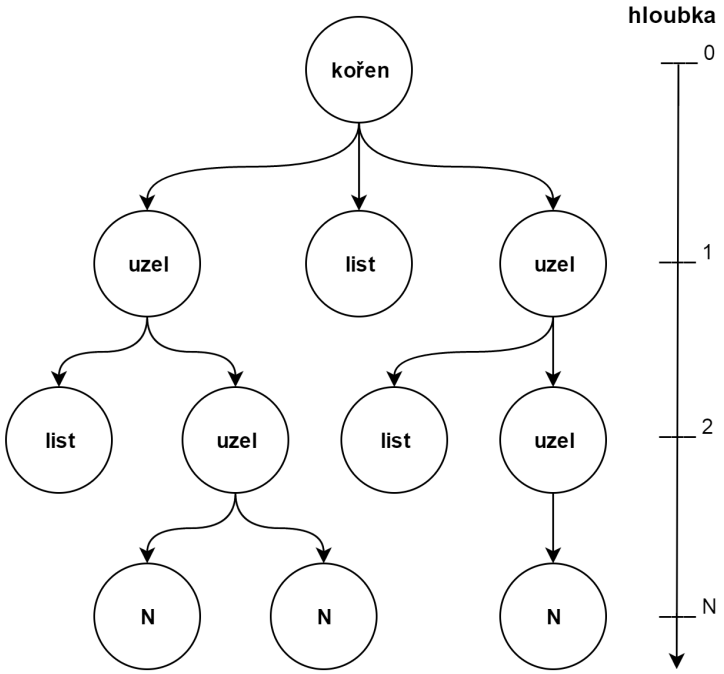
2.2.2 Abstraktní struktura archivačního formátu

Struktura *archivačního formátu* pro webové stránky musí kopírovat základní strukturu statických a dynamických webových stránek. Každá webová stránka je navržená pomocí *stromové struktury stránky*, která je specifikována pomocí *Document Object Model*, dále jen DOM[7]. DOM definuje jednotný přístup pro práci se stromem a jeho částmi.

První část je známa jako kořen, který tvoří výchozí bod stromové struktury. Další částí je uzel, který je rozšiřujícím bodem a obsahuje určitou informaci. Rozšiřující bod znamená, že může mít své potomky, a to jak další uzly, tak listy. Třetí část stromové struktury se nazývá list, stejně jako uzel je nositelem informace, ale již nemá další potomky. Poslední částí je hrana či vazba a slouží k propojení uzlů, listů a kořene[7].

Ve struktuře archivačního formátu bude kořen brán jako vstupní bod, který je často interpretován kořenovým hypertextovým dokumentem. Uzly budou hypertextové dokumenty, které odkazují na další objekty. List je dokument, který již neodkazuje na další objekt. Obrázky či videa budou často interpretované jako list, protože nemají dalšího následovníka. Soubory v archivu jsou mezi sebou propojené pomocí tzv. odkazů. Odkazy jsou z hlediska stromové struktury brány jako vazby.

Na obrázku 2.8 je zachycen graf abstraktní stromové struktury. Takový strom je charakteristický díky rozdělení do úrovní. Počet objektů v jednotlivé úrovni je ovlivňován pomocí počtů vazeb vytvořených z objektů o úroveň vyšší. Základní vlastnost stromové struktury je hloubka, která určuje míru zanoření, tedy počet úrovní stromu. Na vizualizaci grafu je strom zobrazen se čtyřmi úrovněmi, kde poslední úroveň tvoří N. Tento počet je ovlivněn tvorbou vazeb u objektů předešlé úrovně. Objekt, který vytváří vazbu, je nazýván jako rodičovský a je popsán pomocí uzlu. Odkazované objekty se nazývají potomci a mohou vytvářet další vazby. V případě, že potomek vazby vytváří, je chápán opět jako rodičovský objekt pro další úroveň potomků. Jestliže potomek vazby na další objekty nevytváří, je popsán pomocí listu. Na nulté úrovni, též kořenové, se nalézá kořen stromu, který je rodičovským objektem a odkazuje pomocí vazeb na uzly či listy.



Obrázek 2.8: Zobrazení základní stromové struktury

V případě, že stromová struktura je aplikovaná na webovou stránku, obsahuje kořen stromu hypertextový dokument HTML. Dokument vytváří vazby na další textové či binární dokumenty. Textový dokument, který obnáší vazby na další dokumenty, je popsán ve stromové struktuře jako rodičovský objekt. Pokud žádné vazby neobnáší je chápán pouze jako potomek. Druhý typ dokumentu je binární, například obrázek, a díky jeho vlastnostem je vždy popsán jako potomek. Pomocí těchto vazeb lze sestavit stránku do výsledné podoby.

Obsah kořenového dokumentu webové stránky je typicky zapsán pomocí tzv. značek. Pro zápis pomocí značek je navržen značkovací jazyk, který se nazývá *HyperText Markup Language*, dále jen HTML [5]. V HTML dokumentu se lze pomocí speciálních značek odkazovat na další hypertextové dokumenty. Tyto dokumenty mohou být jak binární, tak textové. Textové dokumenty můžou obsahovat další odkazy a jejich obsah je často používaný pro uložení vizualizací stránek nebo pro interpretaci skriptů. Pokud chce data správně interpretovat a zobrazit, musí každý archivační formát podporovat stromovou strukturu webové stránky a její obsah.

2.3 Archivační formát MHTML

Archivační formát *MIME HyperText Markup Language*, dále jen MHTML ukládá jeden dokument typu HTML a ty jeho potřebné zdroje, které jsou reprezentovány pomocí obrázků, hudby či externích odkazů [17]. Formát je využíván k archivaci jediné webové stránky pro možnost jejího pozdějšího zobrazení. Stránka je v archivačním formátu uložena jako otisk, který je daný okamžikem uložení stránky. Objekty jsou uložené v MHTML archivu pomocí rozšíření MIME popsané níže, které umožnuje uložit více různých druhů hypertextových dokumentů v jakémkoliv formátu. Podpora MHTML formátu je zabudovaná v každém moderním prohlížeči. Při použití na dynamických webech archivační formát nemusí správně pracovat s dynamickými skripty, kde nelze jednoduše interpretovat výsledek jako otisk jednoho okamžiku. Možnosti archivačního formátu jsou pevně vázané na rozšíření MIME.

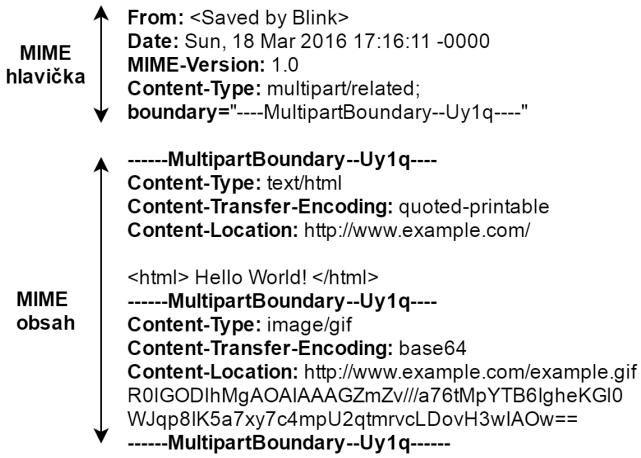
2.3.1 Formát uložení MIME

Rozšíření *Multipurpose Internet Mail Extensions*, dále jen MIME, umožňuje posílat hypertextové dokumenty, binární soubory, kódovaný obsah, či vícejazyčné texty v nejrůznějších formátech [12]. MIME je specifikovaný pěti standardy RFC 2045-2050. Rozšíření se používá pro přenášení příloh a zpráv v rámci poštovní komunikace. Využívá se i při stahování či nahrávání souborů skrze protokol HTTP. Slouží také jako formát pro ukládání objektu ve zmíněném MHTML formátu. Rozšíření MIME, také známo jako MIME zpráva, se dělí na hlavičku a tělo neboli obsah zprávy. Díky jejich využití v poštovní komunikaci obsahují informace v hlavičce verzi, čas poslání či vytvoření zprávy, adresáta, předmět zprávy a typ rozšíření.

Podle typu rozšíření v hlavičce se pozná, jestli je zpráva či archiv jednodílný nebo vícedílný. Vícedílné rozšíření je specifikováno pomocí hodnoty *multipart* v typu rozšíření uvedené v hlavičce. Vícedílné rozšíření se dělí na čtyři základní typy *multipart/message*, *multipart/-mixed*, *multipart/alternative* a *multipart/related* [12]. Pro rozeznání obsahu zprávy je každá taková zpráva v poštovní komunikaci je definovaná podle typu *message*. Typ *mixed* je využíván při posílání obsahu objektů pomoci různorodých hlaviček obsahu. *Alternative* typ umožňuje ke každému objektu přiřadit objekt v alternativním tvaru. Využívá se například pří posílání prostého textu v poštovní zprávě a HTML dokumentu jako alternativy. Tento typ se může často objevovat v hlavičce objektu v obsahové části s kombinací *mixed* v hlavičce zprávy. Poslední typ je *related*, který definuje, že struktura obsahových hlaviček je stejná. Typ je definován ve standardu MHTML pro jeho využití [17]. Každý soubor má totiž stejně definovanou obsahovou hlavičku, tzn. se stejnými parametry.

Typ MIME *multipart/related* je zachycen na obrázku 2.9. Každý MHTML využívá zápis pomocí této struktury. První část MIME rozšíření tvoří hlavička, přičemž její parametry byly představeny v předešlém odstavci. Parametr, respektive odesílatel, je vyplněn názvem programu, který vytvořil archiv. Prohlížeč vytvářející archiv je Google Chrome[6], který využívá jádro *Blink*[7]. Ostatní prohlížeče vloží do odesílatele název svého jádra. Parametr datum říká, kdy byl vytvořen otisk webové stránky a uložen do archivačního formátu. Verze a typ MIME rozšíření jsou dané exportem do archivu.

[6] Chrome - https://www.google.com/chrome/browser/desktop/index.html
[7] Blink - http://www.chromium.org/blink



Obrázek 2.9: Zápis rozšíření MIME *multipart/related*

Kromě popsaných parametrů se nalézá v hlavičce také parametr určující hranice jednotlivých dokumentů s názvem *boundary*. Tento parametr pak umožňuje ukládat více typů hypertextových dokumentů do obsahové části a definuje hranice hypertextových dokumentů pomocí speciálního oddělovače uvedený jako hodnota parametru. Oddělovač je dělicí hranice každého uloženého hypertextového dokumentu. Na obrázku je *boundary* oddělovač definován jako sekvence znaků „*----MultipartBoundary--Uy1q----*“.

Obsahová část se dělí na samostatné hypertextové dokumenty. Tyto části se také někdy nazývají jako *boundary parts*. Jakmile se objeví *boundary* oddělovač na samostatném řádku, značí to začátek načítání nového hypertextového dokumentu. Po oddělovači následuje obsahová hlavička, ve které jsou definované parametry dokumentu. Každá obsahová hlavička má definované parametry jako typ dokumentu, tedy *content-type*, kódování obsahu dokumentu, tedy *content-transfer-encoding*, a umístění dokumentu, tedy *content-location*. Dalším prvkem je sekvence znaků mezi obsahovou hlavičkou a obsahem dokumentu. Tato sekvence je interpretována znaky CLRF, neboli prázdným řádkem, a signalizuje počátek obsahu dokumentu. Poté následuje obsah dokumentu. Ten pokračuje do objevení dalšího *boundary* oddělovače či ukončení MIME rozšíření. Ukončení rozšíření je rovněž definováno pomocí *boundary* oddělovače, a navíc na svém konci obsahuje dva další znaky „**--**“. Díky této posloupnosti lze rozeznat ukončení rozšíření, stejně tak i ukončení archivačního formátu MHML.

2.3.2 Struktura MHTML formátu

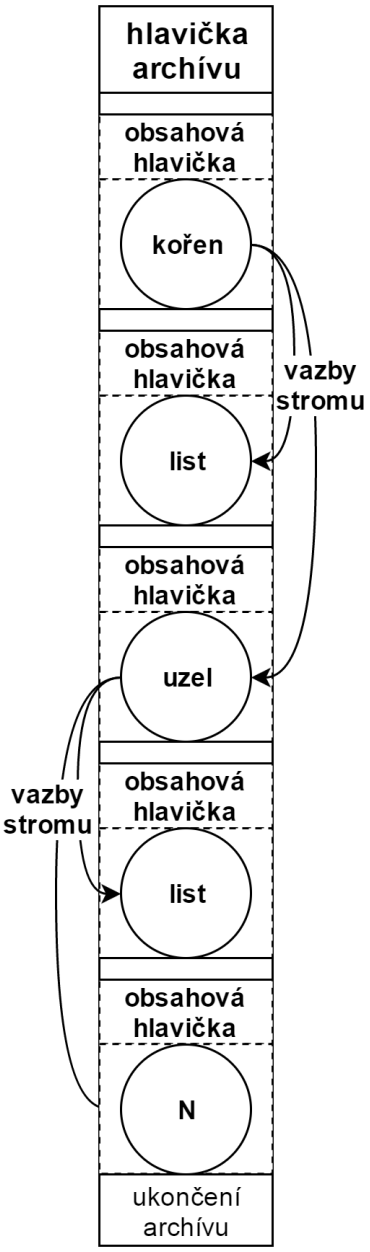
Podstata uložení vícenásobných hypertextových dokumentu v archivačním formátu MHTML, jejich datová interpretace a způsob oddělení dokumentů je vysvětlen pomocí *multipart/related* MIME struktury tvořící opěrný bod formátu. Celkovou strukturu archivu je třeba začlenit do obecné roviny archivačních formátů. Jak bylo uvedeno výše, archiv musí implementovat stromovou datovou strukturu pro uložení a zobrazení dat.

V případě MHTML archivačního formátu je datová struktura udána ve formě jednosměrné vázaného seznamu, kdy je směr udán od prvního prvku po poslední prvek. Seznam lze popsat jako jednodušší variantu stromové datové struktury, která je potřeba pro správnou interpretaci webových stránek. Díky této struktuře lze jednoduše přetransformovat jednosměrný vázaný seznam z grafu na obrázku 2.9 na stromovou datovou strukturu.

Stromová struktura je znázorněna grafem na obrázku 2.10. Základní rozdělení hlavičky a těla MIME rozšíření zůstává. Nový prvek (dokument) seznamu vždy začíná oddělovačem, ten je na grafu znázorněn jako dělicí prvek mezi obsahem dokumentu a další hlavičkou dokumentu. Každý objekt stromové struktury se nalézá v obsahu dokumentu. Dokumenty jsou interpretovány na základě jejich vlastností do podoby objektů stromové struktury, tedy kořene, uzlu či listu.

Kořenový objekt stromu, stejně jak u webové stránky, tvoří HTML dokument, který se váže na další dokumenty. Dokumenty a jejich vazby za sebou nemusí bezprostředně následovat. Mezi těmito dokumenty se pak může objevit libovolný počet dalších dokumentů propojených skrze vazby z předchozích uzlů. Za jedinou pevně specifikovanou podmínku lze považovat pořadí, tedy směr vázaní. Prakticky vzato tak před sebou musí mít jakýkoliv odkazovaný dokument, tedy potomek, již definovaného předka čili rodiče.

Při počátku načítání MHTML archivačního formátu se přečte jednosměrný vázaný seznam. Tak se získají všechny dokumenty v podobě vhodné pro interpretaci dat a zobrazení v prohlížeči či jiné aplikaci. Poté se pomocí kořenového HTML dokumentu vytvářejí vazby na další získané dokumenty. Postupně se skládá stromová struktura do koncového stavu, umožňující výslednou manipulaci pro správnou interpretaci a zobrazení v podobě webové stránky.



Obrázek 2.10: MHTML z pohledu stromové struktury

2.4 Mozilla Archive File Format

Mozilla Archive File Format, dále jen MAFF či MAF formát, je archivační formát pro ukládání jedné či více webových stránek, obrázků nebo jakéhokoliv jiného stahovatelného obsahu. Dokáže uložit informace jako odkaz na originální stránku, kdy a jakým způsobem byl archiv vytvořen či pořídit otisk, tzv. *snapshot*, daného okamžiku ukládané webové stránky. MAFF formát byl vytvořen jako projekt, který implementuje rozšíření, tzv. *plugin*[8], do prohlížeče Mozilla Firefox[9]. První verze formátu byla vytvořena počátkem května roku 2004. Autoři formátu jsou Christopher Ottley a Paolo Amadini [15].

2.4.1 Charakteristika formátu

Projekt je společně se svou dokumentací uložen na platformě *Mozdev*. Tato platforma zadarmo umožňuje místo pro hostování, tedy vývoj a uložení, pro všechny Mozilla aplikace a rozšíření [3]. Mozilla je ochranná známka neziskové organizace Mozilla Foundation, která vyvíjí *open source* projekty jako Mozilla Firefox či Mozilla Thunderbird. MAF formát také spadá pod *open source* díky GPL 3.0[10]licenci.

Základní vlastnosti MAF formátu jsou definované z dokumentace projektu [15]. Pro lepší pochopení formátu je potřeba popsat nejvýznamnější charakteristiky formátu. Svou základní charakteristikou umožnuje uložit veškerý obsah jedné či více webových stránek do jednoho archivu. Každý archiv využívá komprimační technologii ZIP formátu. ZIP formát je exportním formátem umožňujícím bezztrátovou datovou kompresi [13]. Jeho typické koncovky jsou „.zip“ nebo „.zipx“. ZIP formát je definován pomocí specifikace ZIP File Format Specification[13], kterou vyvinula firma PKWARE[11]. Pro zobrazení ZIP formátu se využívá JAR protokol[12], který umožňuje zobrazení ZIP (MAFF) formátu v prohlížeči. JAR neboli *Java Archive* zavádí stromovou strukturu do komprimovaného souboru. Díky stromové struktuře implementované v JAR, dokáže prohlížeč sestavit stromovou strukturu webové stránky, a tu poté zobrazit.

Mezi další charakteristiky MAF formátu patří uložení videí, hudby a jiných zvukových souborů. Tuto vlastnost například postrádá MHTML dokument. Díky tomu je pak MAF formát používán pro ukládání veškerých video nahrávek či složitějších vizualizací. Rovněž také umožňuje ukládání dynamických webových stránek skrze dva základní způsoby. První z nich funguje na principu uložení dat v takovém stavu, v jakém je prohlížeč obdrží, zatímco druhý způsob uložení funguje na principu vytvoření otisku webové stránky z okamžiku ukládání. Přístupy budou blíže popsány v další sekci.

2.4.2 Základní struktura formátu

Základní struktura MAF formátu je definována pomocí MAFF specifikace[16]. Pokud není uvedeno jinak, jsou informace ke struktuře MAF formátu získány z vlastní analýzy. Základem struktury formátu je komprimovaný soubor rozdělený do tří základních komprimačních úrovní. První neumožňuje žádnou komprimaci dat, je tedy vzhledem k paměťové velikosti největší. Výhodou je nejrychlejší zobrazení výsledných webových stránek z ostatních tří

[8] Plugin - https://en.wikipedia.org/wiki/Plug-in_(computing)
[9] firefox - https://www.mozilla.org/cs/firefox/new/
[10] GPL3.0 - http://www.gnu.org/licenses/gpl-3.0.en.html
[11] PKWARE - https://www.pkware.com
[12] JAR protocol - https://docs.oracle.com/cd/E19253-01/819-0913/author/jar.html

úrovní. Další úroveň umožňuje základní datovou kompresi, ta odpovídá standardnímu nastavení ZIP formátů. Poslední úroveň slouží k maximálně možné datové komprimaci, která má bohužel za následek pomalé načítání dat při zobrazení. Komprimovaný soubor je vzhledem k vnitřní struktuře brán jako kořenový adresář.

MAFF využívá adresářovou strukturu, kde musí být kořenový adresář vždy prázdný[16]. To znamená, že nesmí obsahovat žádný dokument či soubor, pouze jen další adresáře. Adresářová struktura vychází ze stromové struktury určené pro uložení souborů či adresářů v rámci souborového systému. Adresářová struktura uvnitř formátu je důležitou komponentou pro uložení a zobrazení více webových stránek.

V MAF formátu je kombinace dvou typu stromových struktur. První z nich je *adresářová struktura*, ta obsahuje vazby na jednotlivé adresáře uložených webových stránek. Druhá stromová struktura webové stránky, ta je naopak důležitá pro sestavení výsledné jednotlivé webové stránky. Pro lepší kategorizaci těchto dvou odlišných typů se bude strom určený pro ukládání v rámci adresářů a souborů stále nazývat adresářová struktura, zatímco pro ukládání webové stránky bude zaveden termín *stromová struktura stránky.*

Adresáře na první úrovni *adresářové struktury* definují samostatné části, které jsou interpretované jako webové stránky. Části se poté zobrazí jako samostatná záložky v prohlížeči. Adresář na první úrovni by měl obsahovat *index.rdf* soubor, kde jsou uložena metadata[16]. V Případě, že adresář neobsahuje *index.rdf*, musí být kořenový HTML dokument pojmenován *index.htm* nebo *index.html*, jinak je archiv neplatný. Obsahuje-li adresář *index.rdf*, metadata v něm definují jméno kořenového HTML dokumentu. V adresáři na první úrovni můžou být maximálně jen soubory *index.rdf*, kořenový dokument, nejčastěji *index.hml*, a další adresář s objekty kořenového dokumentu.

Soubor *index.rdf* musí definovat metadata následující typu; jméno hlavního dokumentu a originální URL stránky, ze které se ukládalo. Dále musí obsahovat datum a čas uložení, titulek webové stránky a kódování HTML dokumentu. Další vlastnosti jsou volitelné a ve specifikaci nejsou uvedeny.

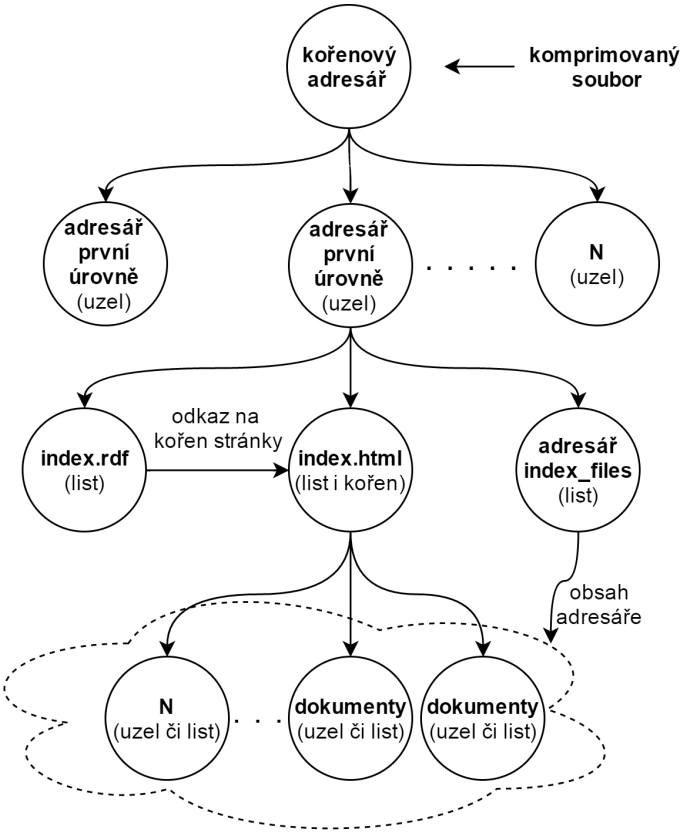
```
<?xml version="1.0"?>
<RDF:RDF xmlns:MAF="http://maf.mozdev.org/metadata/rdf#"
      xmlns:NC="http://home.netscape.com/NC-rdf#"
      xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <RDF:Description RDF:about="urn:root">
  <MAF:originalurl RDF:resource="http://www.example.com/"/>
  <MAF:title RDF:resource="Example Domain"/>
  <MAF:archivetime RDF:resource="Mon, 11 Jan 2016 20:03:33 +0200"/>
  <MAF:indexfilename RDF:resource="index.html"/>
  <MAF:charset RDF:resource="UTF-8"/>
 </RDF:Description>
</RDF:RDF>
```

Obrázek 2.11: Soubor s metadaty *index.rdf* a jeho struktura

Struktura *index.rdf* je zachycena na obrázku 2.11. Struktura je definovaná pomocí *Extensible Markup Language*, dále jen XML, založené na značkovacích jazycích [8]. XML dokáže pomocí jednoduchých značek sdělit informace pro definované struktury skrze jejich odpovídající značku. Díky tomu *index.rdf* definuje struktury, které popisuji vlastnosti archivu. Na obrázku lze vidět již zmíněné struktury, například URL, titulek, vytvoření archivu, jméno kořenového hypertextového dokumentu a jeho znakovou sadu. Soubor *index.rdf* obsahovat může také rozšiřující struktury jako specifikaci formátu a jeho závislosti.

Hypertextový dokument nalézající se v adresáři první úrovně obsahuje základní informace webové stránky, včetně odkazů na další dokumenty potřebné k sestavení. Další hypertextové dokumenty odkazované z kořenového dokumentu jsou obsaženy v definovaném adresáři pod názvem *index_files*. V adresáři *index_files* je stromová struktura dalších adresářů a dokumentů různorodá, přičemž není nijak omezena. Stromová struktura většinou kopíruje původní strukturu ukládané webové stránky.

Popsané informace MAF formátu jsou zachyceny grafem na obrázku 2.12. Graf obsahuje abstraktní vícenásobnou stromovou strukturu MAF formátu. Počáteční bod vnitřní struktury archivu tvoří kořenový adresář, ten tvoří kořen adresářového stromu. Adresáře na první úrovni odpovídají vlastnostem uzlu interpretovaného ve vizualizaci jako záložka. Každý uzel na sebe váže potomky, jejichž vlastnosti odpovídají rysům listu stromu. Hloubka *adresářové struktury* díky ukončení stromu listy nikdy nemá hloubku větší než tři. Důležitý je potomek, který je interpretován hypertextovým dokumentem *index.html*, poněvadž tvoří kořen *stromové struktury stránky*. V případě stromové struktury stránky jsou všechny úrovně stromu, kromě kořenové, uloženy v adresáři *index_files*. Stromová struktura stránky na sebe váže všechny obsažené dokumenty v adresáři *index_files*. Tato struktura nemá definovanou konečnou hloubku stromu, její abstraktní návrh vychází ze základní struktury stromu zobrazený na obrázku 2.8.



Obrázek 2.12: Abstraktní struktura MAF formátu

Načítaní MAFF archivu pro zobrazení obsahuje výčet akcí tvořících výslednou vizualizaci. Z analýzy Firefox[13]prohlížeče a JAR protokolu vyplývá, že v prvním kroku se pomocí adresářové struktury MAF formátu vytvoří záložky v prohlížeči. Poté se pro každou jednotlivou záložku pomocí metadat vyhodnotí kořenový dokument *stromové struktury stránky* a jeho znaková sada. Kořenový dokument se předá do jednotlivé záložky a odkazované dokumenty se začnou vázat ke kořeni stromu a sestavovat výslednou podobu stránky. Je nutno zdůraznit, že v této fázi se každý soubor získávaný z MAF formátu musí nejdříve dekomprimovat a poté se vkládá do výsledného stromu. Jakmile je strom v cílové podobě, stránka v záložce se zobrazí a přechází se na další záložku. V prohlížeči Firefox lze i definovat, zdali se záložky mají skládat sekvenčně či paralelně. V případě sekvenčního skládání se záložky vytvářejí a načítají postupně, zatímco paralelně se záložky načítají v jediném čase. Tato vlastnost je již definovaná v prohlížeči a formát ji neovlivňuje.

2.4.3 Metody uložení v rámci formátu

Další důležitou vlastností MAF formátu je dělení podle uložení stromové struktury ukládané stránky. Tato vlastnost je sice interpretována prohlížečem, ale formát dokáže podporovat oba přístupy i navzdory velkým zobrazovacím rozdílům.

První technika se nazývá *browser saving system* a využívá originální stromovou strukturu prohlížeče, ze kterého je archiv vytvořen. To znamená, že hypertextové dokumenty jsou v podobě, jak je klient (prohlížeč) obdržel od webového serveru. Kromě standardních hypertextových dokumentů technika ukládá prvky, jako jsou například *cookies*, dynamické požadavky a odpovědi, skriptovací dokumenty v původní podobě a další prvky. Výhoda techniky spočívá v obsažení veškerých informací od prvotního zobrazení stránky až po konečné zobrazení stránky ovlivněné uživatelskou interakci. V praxi se bohužel jedna spíše o nevýhodu, standardní chování při zobrazení formátu totiž zobrazí pouze informace uložené v jediném okamžiku. V takovém případě je webová stránka nevalidně zobrazena, protože veškeré informace a dokumenty z většího časového celku se zobrazují v jeden okamžik. Jestliže bude vytvořena aplikace pro zobrazení stránky obsahující pokročilou heuristiku, bude schopna zobrazit celou časovou osu stránky skrze takové informace.

Druhá technika využívá principu podobných jako v MHTML a jmenuje se *snapshot*. Z definice vyplývá, že technika vytvoří otisk webové stránky pouze v době jejího ukládání, a proto zde neexistuje možnost zobrazení stránky v předchozí době. Technika totiž funguje na principu získávání informací ze skriptovacích dokumentů pouze během jednoho okamžiku. Na rozdíl od techniky předešlé nejsou ve stromové struktuře stránky uloženy skriptovací dokumenty. Tyto dokumenty jsou totiž vyhodnocené prohlížečem při zobrazení pomocí uživatelské interakce. Ve chvíli vzniku archivu se z prohlížeče převezme dynamicky vygenerovaný HTML dokument. Dokument obsahuje informace vygenerované ze skriptovacích dokumentů, díky tomu již další části skriptů není třeba. Velkou výhodou této techniky je zobrazení stránky ve stejném stavu jakém byla při ukládání. Zobrazování dat je i oproti technice *browser saving system* rychlejší, protože nemusí řešit načítání dynamických dokumentů. Jako nevýhodu však archiv neobsahuje veškeré informace o proměně stránky. Tyto informace ale obyčejný uživatel obvykle nepotřebuje, tato nevýhoda je tedy bezvýznamná oproti benefitům získaným díky *snapshot* technice.

[13]https://hg.mozilla.org/mozilla-central/file/1152d99d8c53

Kapitola 3

Návrh a implementace modulu pro rekonstrukci webu

Cílem této kapitoly je seznámení s návrhem implementace pro zpětnou rekonstrukci webového provozu, včetně realizace rekonstrukce a to pomocí implementovaného modulu pro cílový nástroj *Netfox Detective*. Kapitola se dále zaměří na cílovou platformu, využití již existujících modulů v této platformě a výstupní formáty potřebné pro realizaci modulu.

Kapitola také popisuje potřebné technologie a jejich vlastnosti ovlivňující uložení či zobrazení rekonstruovaných webových stránek. Na závěr kapitoly je uvedena implementace základních částí modulu. Ta popisuje vytváření archivačního formátu z webového toku, jeho proces uložení a výsledné zobrazení přímo prostřednictvím modulu.

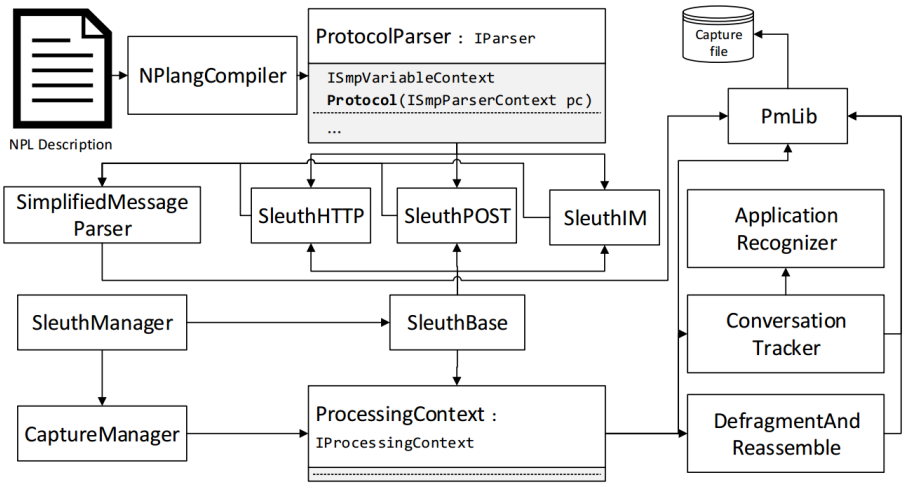
3.1 Analýza požadavku na návrh řešení modulu

Z analýzy požadavků na vytvoření modulu pro zpětnou rekonstrukci webového provozu vyplývají náležitosti, které musí modul splňovat. První náležitostí je implementace modulu pro nástroj *NetFox Detective*. Druhou je uložení zrekonstruované webové stránky a jejího obsahu do výsledného *archivačního formátu* MAFF. Díky tomuto *archivačnímu formátu* lze webové stránky a jejich obsah kdykoliv znovu zobrazit či použít pro další forenzní analýzy. Třetí náležitostí implementačního návrhu je implementace vizualizace uloženého webového obsahu přímo v navrženém modulu. Technologie navržené pro vizualizaci obsahu v rámci modulu jsou uvedené v oddílu 3.1.3. Poslední náležitostí návrhu jsou pak souhrnné statistiky, včetně výčtů hypertextových dokumentů pro každý výstupní MAFF archiv.

3.1.1 Nástroj pro analýzu Netfox Detective

Netfox Detective dále jen NFX, je nástroj umožňující síťovou forenzní analýzu. Nástroj byl vytvořen jako volně dostupná platforma poskytující tzv. *conversation-based* přístup umožňující pokročilé dolování dat ze zachycené síťové komunikace. NFX dokáže sestavovat jak TCP/IP provoz, tak i jiné mechanismy, jako jsou například tunelovací protokoly. Pro základní pochopení nástroje *Netfox Detective* je na obrázku 3.1 zachycena jeho abstraktní architektura [18].

Hlavním exportním výstupem NFX jsou konverzace, které odpovídají jednotlivému TCP spojení. Každá konverzace je uložena pro pozdější použití a obnáší metadata jako zdrojovou a cílovou IP adresu, zdrojový a cílový port nebo čas zachycení konverzace. Díky metadatům lze jednotlivé uložené konverzace dále analyzovat prostřednictvím rozšiřujících modulů.



Obrázek 3.1: Architektura nástroje Netfox Detective [18]

3.1.2 Rozšiřující moduly a modul pro exportování webových objektu

Důležitou součástí NFX jsou rozšiřující moduly, tzv. *sleuths*, umožňující interpretovat konverzace získané z dolování dat. Každý modul umožňuje rozšířit funkcionalitu získaných konverzací ve formě sémantického významu, nejčastěji ve formě aplikačního protokolu. Moduly umožňují flexibilně rozšířit forenzní analýzu a to ve formě analýzy dalších síťových protokolů [18].

Mezi základními rozšířeními nalezneme HTTP modul (*sleuth*), který rozšiřuje forenzní analýzu pro HTTP protokol a vytváří ze zachycených konverzací hypertextové dokumenty. Každý takový hypertextový dokument je tvořen pomocí požadavku a odpovědi (konverzace). Při zpracování konverzace se nejdříve přečte hlavička požadavku, kde je uvedena cesta a název dokumentu. Poté se zpracuje odpověď obsahující náležitosti požadovaného dokumentu. Získané dokumenty se po dokončení zpracování uloží a vytvoří se metadata popisující jednotlivé hypertextové dokumenty. Tyto dokumenty tvoří výstupy z HTTP modulu, které lze pomocí jejich metadat použít jako vstup pro další forenzní analýzu.

Modul pro exportování obsahu webového provozu do MAF formátu, dále jen MAFF modul, je implementován jako další rozšiřující modul pro NFX. Modul je navržen, aby využíval jako svůj vstup exportované hypertextové dokumenty z HTTP modulu. Tyto dokumenty se zpracují do formátu potřebného k uložení do MAFF archivu. Každý vytvořený archiv je výstupní entitou MAFF modulu obsahující exportovanou webovou stránkou. Modul umožňuje každý exportovaný archiv zobrazit přímo v *Netfox Detective* a také zobrazuje souhrnné údaje o exportovaném archivu.

3.1.3 Technologické požadavky na MAFF modul

Požadavky na MAFF modul ovlivňují celý jeho návrh i realizaci a jsou jeho základní charakteristikou. Prvním požadavkem modulu je programovací jazyk, ve kterém je MAFF modul implementován. Vybraným programovacím jazykem je C#, pomocí kterého je také implementován *Netfox Detective*. Jazyk C#, celým názvem *Visual C# Language*, je novodobý, imperativní, objektově orientovaný programovací jazyk navržen pro vytváření komplexních aplikací, které běží na *.NET Frameworku* [9].

Dalším požadavkem modulu je vytvoření MAFF archivů ze vstupních dokumentů a jeho uložení ve formě výstupního exportu. Technologický požadavek při vytváření exportního MAFF archivu je jeho komprimace, která je popsaná v oddílu 2.4.1. Standardní technologie datové komprimace definovaná PKWARE specifikací je obsažena v technologií *.NET Framework*. Třída .NET platformy popisující datovou komprimaci se nazývá ***ZipArchive****(filestream, ziparchive mode)*[1]a jejími parametry jsou otevřený soubor a mód použití archivu. Módy použití jsou *read*, *create* a *update*. První dva módy archivu slouží jen pro čtení nebo zápis, zatímco mód *update* slouží jak pro čtení, tak pro zápis.

Předposledním požadavkem, který musí modul splňovat je zobrazení MAFF archivu přímo v *Netfox Detective*. Ačkoliv je podle MAFF standardu definováno zobrazování MAFF archivu přes JAR protokol popsaný v oddílu 2.4.1, nelze z důvodu implementačního jazyka C# tuto technologií použít. Technologie zobrazování pomocí JAR protokolu je totiž závislá na programovacím jazyce Java. K zobrazení archivů je tedy využita alternativní technologie *Chronium Embedded Framework for CSharp*, dále jen *CefSharp*, vycházející z volně dostupného prohlížeče *Chronium*[2]. *CefSharp* popisuje *Chronium* prohlížeč integrovaný do *.NET Framework* zobrazovací technologie *Windows Presentation Foundation*[3], dále jen WPF [1]. Zobrazovací technologie vytváří grafické rozhraní ve formě okna (window) pro zobrazení a vykreslení požadovaných informací. Bezpečnostní nástroj *Netfox Detective* využívá technologií WPF pro zobrazování exportních výstupu. Z tohoto důvodu je rovněž použitá v MAFF modulu. WFP technologie využívá k propojení grafických prvků okna a dat vazbu nazývající se *data binding*[4]. Vazba *data binding* umožňuje jednoduchou interakci a manipulaci dat prostřednictvím uživatelských akcí. *CefSharp* využívá tuto vazbu při načítaní požadovaného obsahu webové stránky a díky této vazbě vykreslí načtený obsah do WPF okna. Nevýhodou této alternativní technologie je neschopnost zobrazit data přímo v komprimovaném formátu, a proto je nutné hypertextové dokumenty vždy před zobrazením stránky dekomprimovat.

Posledním základním požadavkem modulu je vykreslení a zobrazení souhrnných informací či statistik o každém exportovaném MAFF archivu. Pro zobrazení pohledu souhrnných informací je využita technologie WPF, která umožňuje zobrazení obsahu každého hypertextového dokumentu. Tento pohled také umožňuje otevření jednotlivého dokumentu a interpretaci jeho výsledného zobrazení podle asociované přípony, například JPEG (obrázek) či MP3 (audio). Pro otevření souboru se používá metoda ***Process.Start****(filename)*, kde její parametr určuje jméno a cestu k souboru. Tato metoda je součástí třídy *Process* integrované do .NET Frameworku.

3.2 Návrh řešení implementace

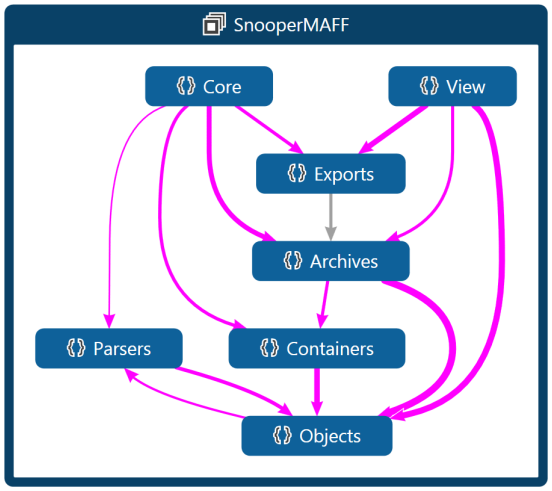
Podkapitola představuje základní návrh implementace a rovněž popisuje některé důležité pasáže implementace MAFF modulu. Samotná realizace modulu podléhá definovaným technickým požadavkům popsané v oddílu 3.1.3. Implementace modulu je pak rozdělena na dvě abstraktní části, které společně tvoří celkovou skladbu MAFF modulu.

[1]ZipArchive - https://msdn.microsoft.com/cs-cz/library/system.io.compression.ziparchive(v=vs.110).aspx
[2]Chronium - https://www.chromium.org
[3]WFP- https://msdn.microsoft.com/en-us/library/ms754130(v=vs.110).aspx
[4]Data binding - https://msdn.microsoft.com/en-us/library/ms752347(v=vs.100).aspx

První část modulu implementuje načítání vstupních hypertextových dokumentů, jejich zpracování do požadovaného tvaru a vytvoření MAFF archivu pro jednotlivé zachycené stránky. Vytvořené MAFF archivy tvoří výstupní export MAFF modulu. Celá tato část je popsána jako *zpracovávání archivu*. Druhá část implementace modulu je zaměřena na vizualizaci výstupních exportních dat MAFF modulu a popsána jako *vizualizace archivu*. Každý jednotlivý obsah výstupního archivu lze pomocí *vizualizace archivu* zobrazit, a to jak vizualizací uložené webové stránky, tak zobrazením souhrnných statistik, včetně obsahu jednotlivého dokumentu v MAFF archivu.

3.2.1 Popis implementace

Implementace modulu je rozdělena na samostatné entity, tedy instance tříd popisující jejich vymezenou funkcionalitu. Rozdělení jednotlivých entit a jejich vzájemná interakce je zachycena na obrázku 3.2. Graf používá anglickou terminologii z důvodu neexistence přesného českého ekvivalentu původního významu objektu.



Obrázek 3.2: Abstraktní schéma modulu pro rekonstrukci dat

Základní entita modulu je *Core*, dále jen jádro. Jádro řídí implementační část *zpracování archivu*, která řeší načítání požadovaných vstupů z HTTP modulu. Část *zpracování archivu* se dělí na základní tři fáze. První se nazývá *fáze zpracování*, kde jádro vytváří entitu *Parsers* sloužící k dalšímu zpracování vstupních HTTP objektů a entitu *Containers* sloužící k uložení zpracovaných objektů. Z důvodu možného výskytu objektu ve více archivech je třeba uložení zpracovaného objektu do pomocného úložiště. Druhá fáze, tj. *fáze skládání*, pak slouží pro skládání objektů pomocí heuristik do cílového archivu. Tato fáze je nejdůležitějším implementačním bodem MAFF modulu. Poslední fáze je *fáze finalizace* a zabývá se kompletací, komprimací a exportem poskládaného archivu.

V první fázi po načtení jednotlivého vstupu jádro předá získaný vstup entitě *Parsers*, která jej přetvoří do objektu odpovídajícímu formátu vhodnému pro uložení do MAFF archivu. Tyto objekty se nazývají MAFF objekty a jsou popsané entitou *Objects*. Po dokončení zpracování se objekty přidají do úložiště, kde čekají na další analýzu. Úložiště je implementováno podle entity *Containers* a je rozděleno na části popisující jednotlivé konverzace. Každá konverzace je popsána pomocí dvojice komunikačních uzlů neboli pomocí klienta a serveru. V případě, že MAFF objekt je objektem kořenovým, není přidán do úložiště, ale vytvoří MAFF archiv sloužící jako vstupní bod pro další fázi. Každý MAFF archiv odpovídá základní entitě *Archives*, která implementuje celkovou strukturu MAFF archivu.

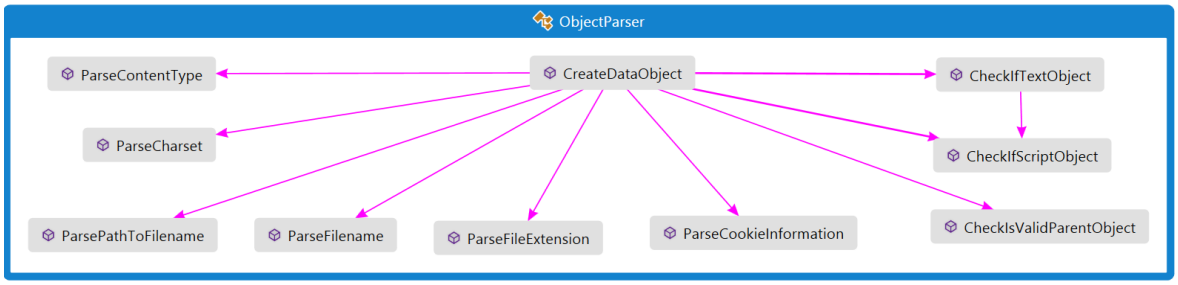
Po dokončení první fáze jádro modulu ukončí zpracovávání objektu a připraví se na *fázi skládání*. V této fázi se postupně prochází vytvořené MAFF archivy, jejichž pomocí jsou vyhledáváni potomci kořenového dokumentu. V případě nalezení jsou takoví potomci přidáni do archivu i s jejich dalšími potomky.

Poslední fáze *zpracování archivu*, tedy fáze finalizace, kompletuje složené archivy, komprimuje je a vytváří metadata popisující archivy jako výstup MAFF modulu. Při kompletaci archivu se podle nastavení MAFF modulu rozhoduje, zdali budou vytvořené snímky *snapshoty* webové stránky či nikoliv. Každý archiv je po kompletaci, komprimaci a vytvoření metadat svými vlastnostmi popsán jako entita *Exports*.

Entita *View* implementuje druhou základní část MAFF modulu, tedy *vizualizace archivu*. V této části se načítá každý archiv ve formě entity *Exports*. Ta pak obsahuje vizualizaci webové stránky či její souhrnné statistiky. Detailnější implementační popis je uvedený v oddílu 3.2.6.

3.2.2 Zpracování vstupních objektů

K získání MAFF objektů v odpovídajícím formátu je třeba upřesnit základní charakteristiky entity *parser* využité ve *fázi zpracování*. Vstupem do této entity je HTTP objekt tvořený požadavkem a odpovědí popsané v oddílu 2.1.2. Jednotlivé části zpracování objektu jsou znázorněné na obrázku 3.3. Každý atribut je získaný skrze odpovídající metodu. Například atribut *ContentType* je získaný pomocí metody *ParseContentType()*.



Obrázek 3.3: Zpracování MAFF objektů

Nejprve dojde ke zpracování hlavičky odpovědi HTTP objektu, kde jsou získány hodnoty atributů pro datový typ (*ContentType*), jeho znakovou sadu (*Charset*) a seznam *Cookies* výstupního MAFF objektu. Hodnota datového typu slouží pro vyhodnocení výstupního MAFF objektu a jeho vlastností v rámci archivu. Druhý atribut určuje prostřednictvím své hodnoty znakovou sadu HTTP objektů. Znaková sada se vyskytuje u textových neboli hypertextových dokumentů. Naproti tomu u binárních HTTP objektů, například obrázku či videí, se nevyskytuje. Hodnota tohoto atributu je velmi důležitá při zpracování kořenového

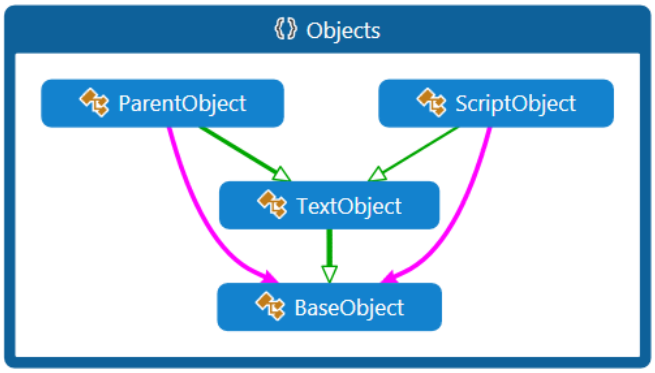
HTML dokumentu webové stránky, poněvadž obvykle určuje i znakovou sadu jeho potomků. Posledním atributem načteným z hlavičky odpovědi je seznam *cookies* využitých při skládání webové stránky v archivu.

Po dokončení hlavičky odpovědi se přechází na zpracování hlavičky požadavku. Při zpracování požadavku se nejprve načte URI adresa z *request line* a uloží se do atributu *OriginalURI* MAFF objektu. Dále se získá cesta k souboru a hodnota této cesty se uloží do atributu *PathToFilename*. Po získání cesty k souboru se zpracuje jméno souboru (*Filename*) a jeho rozšíření (*extension*). Hodnoty těchto atributů se získají z původní hodnoty URI adresy. Dalším atributem získaným z hlavičky požadavku je adresa hosta (*HostAdress*) ze které objekt pochází. Posledním atributem zpracovaným z hlavičky požadavku je *Referrer*. Hodnota tohoto atributu popisuje zdroj, ze kterého se přistupuje na nový cílový zdroj.

Posledním krokem při zpracování HTTP objektu do podoby MAFF objektu je získání obsahu původního dokumentu a vybrání výsledného typu objektu. Obsah MAFF objektu se načítá z těla odpovědi, kde je uložen původní HTTP objekt. Tento obsah je poté interpretován pomocí výběru výsledného typu objektu. K výběru typu výsledného MAFF objektu jsou implementovány metody *CheckIfTextObject()*, *CheckIfScriptObject()* a *CheckIfValidParentObject()*. První metoda ověřuje, je-li výsledným MAFF objektem hypertextový dokument. Druhá pak zjišťuje, jestli objekt obsahuje skript a poslední metoda ověřuje, zdali je kořenový dokument validní. Validní kořenový dokument musí obsahovat HTML a jeho základní značky jako jsou hlavička dokumentu (*<head>*) či tělo (*<body>*). Validní kořenový dokument také musí obsahovat deklaraci stránky tzv. DOCTYPE[5], která slouží pro specifikaci úrovně korektního HTML dokumentu. V případě, že kořenový objekt je validní, je na základě tohoto kořenového objektu vytvořen archiv popisující webovou stránku definovanou kořenovým objektem.

3.2.3 Typy MAFF objektů

Podle vlastnosti MAFF objektu je vybrán výsledný typ, který popisuje charakteristické rysy dané kategorie objektu. Objekt má definované čtyři základní typy a jejich rozdělení je zachyceno na obrázku 3.4.

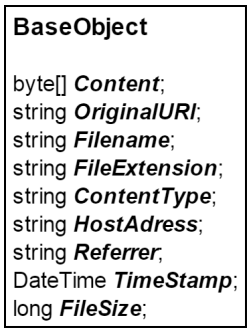


Obrázek 3.4: Rozdělení podle typu objektu

[5]DOCTYPE - http://www.w3schools.com/tags/tag_doctype.asp

Základním typem MAFF objektu je *BaseObject*, dále jen objekt základního typu, popisující základní strukturu MAFF objektu. Objekt základního typu obsahuje atributy potřebné pro zobrazení informací ve vizualizačním pohledu souhrnné statistiky MAFF formátu. Všechny ostatní objekty jsou z něj odvozené a díky tomu lze jakýkoliv typ MAFF objektu zapsat v tomto základním tvaru. MAFF objekty, které zůstanou pouze v základním tvaru, jsou typicky objekty binární, tedy například videa, obrázky či jakýkoliv jiný multimediální obsah.

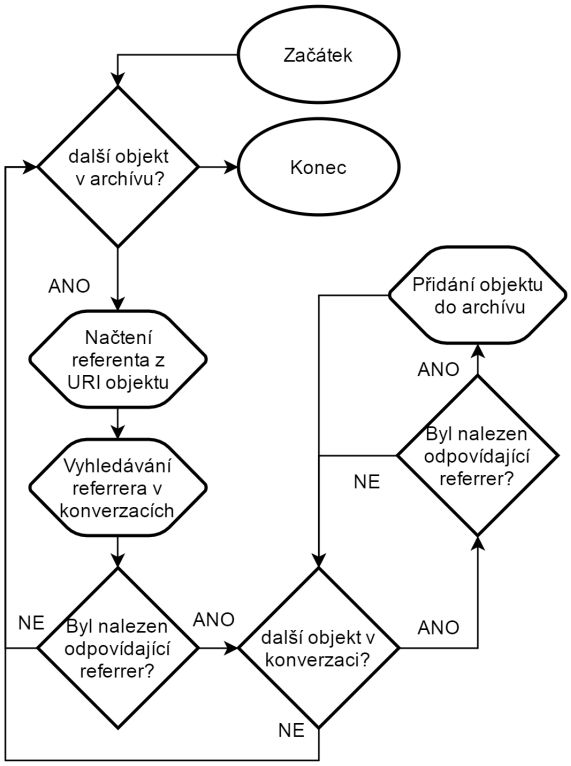
Dalším typem MAFF objektu je *TextObject*, dále jen objekt textového typu, který kromě atributů zděděných ze základního typu objektu má i své rozšiřující atributy. Rozšiřující vlasnosti jsou například reference na další objekty nebo možnost úpravy referencí v obsahu dokumentu při *fázi finalizace*. Objekt textového typu se dále dělí na *ParentObject* a *ScriptObject*. *ParentObject*, dále jen objekt rodičovského typu, je typicky každý validní kořenový dokument jednotlivé webové stránky. V případě, že objekt není v rodičovském tvaru validní, zůstane popsán pouze objektem ve tvaru textovém. *ScriptObject*, dále jen objekt obsahující skript, slouží pro hlubší analýzu a skládání snímků. Na obrázku 3.5 je uveden MAFF objekt v základním tvaru se všemi atributy určenými pro uložení do archivu a v rámci vizualizace souhrnných statistik. MAFF objekt v základním tvaru je požadovaný formát pro ukládání objektu do archivu. Atributy *OriginalURI*, *Filename*, *FileExtension*, *ContentType*, *HostAdress* a *Referrer* byly popsány v oddílu 3.2.2. Atribut *TimeStamp* obsahuje čas, kdy byl objekt zachycen. Atribut *Content* interpretuje obsah MAFF objektu. Atribut *FileSize* udává velikost tohoto obsahu.



Obrázek 3.5: Ukázka základního typu objektu

3.2.4 Skládání MAFF archivu

Skládání archivu se provádí ve *fázi skládání*, která je druhou fází implementační části *zpracovávání archivu*. V této fázi se načítají vytvořené archivy a vyhledávají vazby kořenových dokumentů na další MAFF objekty obsažené v úložišti po dokončení první fáze. Pro vyhledávání, vytváření vazeb a skládání *stromové struktury stránky* v archivu jsou použity základní dvě heuristické metody. První z nich se nazývá *heuristická metoda refererrů*. Ta pro vyhledávání a vytváření vazeb využívá schéma *referrerů* a *referentů*. Při počátku vyhledávání pomocí této heuristiky je vzato originální URI kořenového dokumentu, přičemž této hodnotě se přezdívá *referent*. Poté se začnou vyhledávat konverzace v úložišti, kde každá konverzace má svůj seznam *refererrů*. Pokud je *referrer* v konverzaci nalezen, vyhledají se v rámci konverzace objekt či objekty nesoucí tento *referrer* v hlavičce požadavku a přidají se do výsledného archivu. Tento proces se znovu opakuje pro všechny nově přidané objekty archivu. Algoritmus této heuristické metody je zachycen na obrázku 3.6.



Obrázek 3.6: Heurestická metoda referrerů

Heuristická metoda referrerů je po výkonnostní stránce nenáročná a ve své podstatě tvoří nejrychlejší způsob pro získání objektů a jejich vazeb. Bohužel má tato metoda nízkou úspěšnost nalezení všech objektů v rámci daného archivu.

Prvním důvodem ovlivňujícím neúspěšnost nalezení je možnost, že objekty jsou již načtené v mezipaměti prohlížeče a nemusí být s cílovým archivem propojené pomocí *refererru*. V případě, že uživatel přistoupí na stránku www.example.com, dojde ke stažení hypertextových dokumentů, jako jsou například kaskádové styly. Dokument obsahující kaskádový styl této stránky může být například *view.css*. V případě, že uživatel přistoupí ze stránky www.example.com na stránku www.example.com/docs/myexample.hml, bude HTML soubor *myexample.html* obsahovat *referrera* na stránku www.example.com. V ten moment ale již nový požadavek neobsahuje žádnou návaznost na kaskádové styly *view.css*, a proto se pomocí *heuristiky referrerů* pro tuto webovou stránku dokument již nepřiřadí. HTML soubor *myexample.html* však vazbu na kaskádové styly view.css obsahuje a po vizualizaci archivu by se tak stránka korektně nenačetla.

Druhý důvod ovlivňující neúspěšnost nalezení vazeb na další objekty spočívá ve skutečnosti, že *referrera* není povinno uvádět a někdy je dokonce i nevyžádaný z hlediska bezpečnosti. Proto je potřeba hledat další možnosti skládání stromové struktury stránky v cílovém archivu.

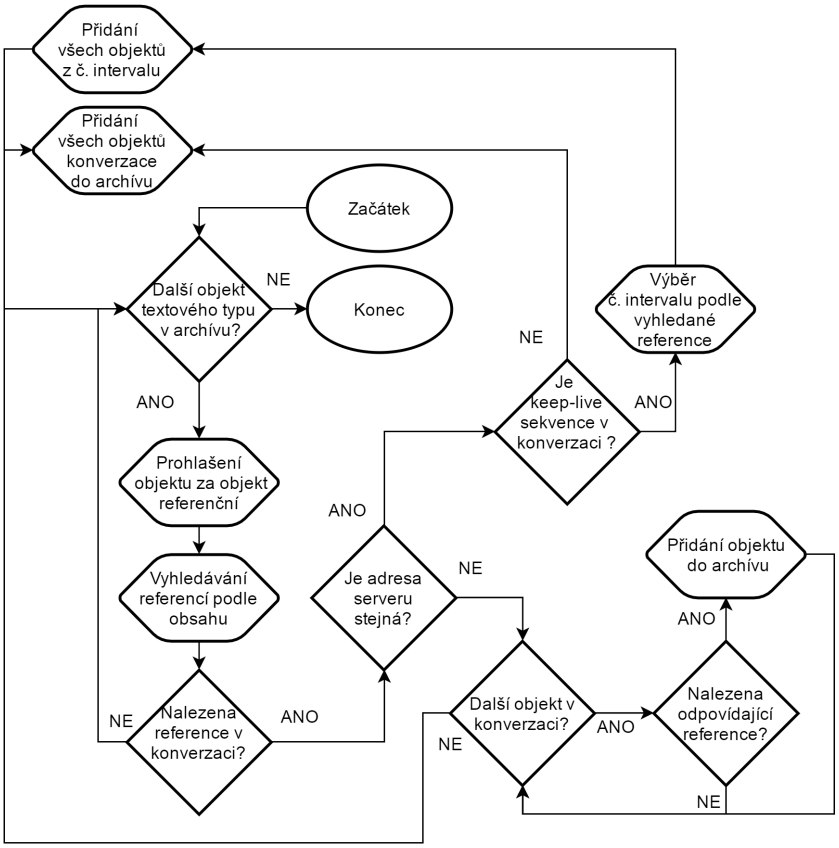
Druhou metodou využívanou modulem MAFF je *heuristická metoda referencí*. Základním prvkem této heuristiky je *reference* neboli vazba na odkazovaný hypertextový dokument či multimediální objekt. *Reference* se vyskytují pouze v hypertextových dokumentech, které jsou definovány na základě MAFF objektu v textovém tvaru. Díky tomu lze každý objekt v textovém tvaru brát jako *objekt referenční*. *Referenční objekt* je následně takový objekt, který může být či přímo je nositelem *referencí* vůči dalším objektům. Díky této charakteristice lze heuristiku optimalizovat pouze na prohledávání objektů v textovém tvaru, které jsou umístěné v MAFF archivu.

Počátkem vyhledávání nových objektů se načte aktuální objekt textového typu obsažený v MAFF archivu a tento objekt se prohlásí za *objekt referenční*. Následně se začne prohledávat úložiště objektů pomocí obsahu *referenčního objektu*. *Reference* vyhledávaných objektů jsou tvořeny skrze atribut*PathToFilename*. Nejprve se vyhledají *reference* pro jednotlivé konverzace obsažené v pomocném úložišti, a pokud konverzace *referenci* obsahuje, vyhledá se v konverzaci daný objekt. Tato část je implementovaná jako dva přístupy s různou časovou složitostí.

První přístup vyhledává objekty buď pouze pro konverzace vlastnící stejnou adresu serveru, (*HostAdress*) jakou má *referenční objekt*. Nebo také hledá v konverzacích, ze kterých již byl přiřazen MAFF objekt prostřednictvím *heuristické metody referrerů*. V případě stejné adresy serveru jsou veškeré objekty obsažené v konverzaci automaticky přiřazené k archivu. Důvodem automatického přiřazení je nemožnost získat objekt z jedné konverzace, který by nebyl součástí archivu. Tento přístup je sice časově náročnější než vyhledávání pomocí *heuristická metoda referrerů*, ale naopak je méně náročný než druhý přístup *heuristické metody referencí*.

Druhý přístup heuristiky nepoužívá žádné omezení vyjma časové značky. Díky tomu je časová náročnost tohoto přístupu sice největší, na druhou stranu je největší i jeho účinnost. Omezení poté existuje ve formě časové značky *referenčního objektu* a zaručuje, že budou analyzovány pouze konverzace trvající maximálně jednu hodinu od časové známky *referenčního objektu*. Dalším faktorem ovlivňujícím *heuristickou metodu referencí* je adresa serveru jednotlivé konverzace. Zde je rovněž uplatněn princip automatického přiřazování objektů jednotlivé konverzace, ale s dodatečným ověřením časového rozestupu v rámci jednotlivé konverzace. Časové rozestupy v konverzaci jsou definovány pomocí *keep-live* sekvence, která rozděluje konverzaci na samostatné celky. Každé vytvořené spojení, které již přeneslo všechny HTTP objekty, ale zároveň nemá být uzavřeno, využívá *keep-live* sekvenci pro udržení životnosti spojení. Tato sekvence se obvykle opakuje každých 10 sekund od posledního přeneseného objektu. Díky této sekvenci tak lze spojení rozdělit na samostatné celky pro *heuristickou metodu referencí*.

Přístup *heuristické metody referencí* je znázorněn na obrázku 3.7. Znázornění je důležité pro pochopení opěrného bodu implementace MAFF modulu.



Obrázek 3.7: Heurestická metoda referencí

Po dokončení vyhledávání *referencí* pomocí obsahu *referenčního objektu* jsou všechny získané objekty přiřazeny do archivu. Zároveň jsou nalezeny všechny nově přidané MAFF objekty textového typu. V případě nalezení objektu textového typu je nový objekt textového typu opětovně vzat a prohlášen za *objekt referenční*. Poté se vyhledávání znovu aplikuje na tento nově získaný *referenční objekt* a algoritmus neskončí, dokud všechny MAFF objekty textového typu nebudou použité pro vyhledávání.

3.2.5 komplementace a komprimace archivu pro výstup

Třetí fáze *zpracovávání archivu*, tedy *fáze finalizace*, je tvořena sestavením a komprimaci MAFF archivu. Sestavení archivu tvoří dvě abstraktní části. První část slouží pro vytváření snímků webové stránky, mají-li se vůbec vytvářet. Druhá část se zabývá přepisováním *referencí* obsažených v MAFF objektech textového typu *referencemi* odpovídajícím stávající hierarchii v rámci ukládaného archivu. Poslední část popisuje vytvoření komprimovaného souboru a uložení MAFF objektů do tohoto souboru. Po dokončení skládání archivu přichází komprimace. Při komprimaci se vytvoří komprimovaný archiv, do kterého se postupně uloží obsah každého MAFF objektu.

V předchozí fázi se při vyhledávání a ukládání nalezených MAFF objektů veškeré objekty ukládaly do abstraktní části archivu pojmenované *archive root part*, tedy kořenové části archivu. V této části jsou uložené MAFF objekty v původním tvaru, jak byly získané ze zachycené komunikace. Tyto objekty jsou ve stejném tvaru, jak popisovala technika uložení *browser saving system* v oddílu 2.4.3. Díky tomu lze zobrazit původní nezměněný obsah MAFF objektů a to umožňuje zobrazení původního vzhledu stránky v nezměněné podobě.

Kořenová část archivu je vstupem při vytváření snímku v rámci skládání archivu. Každý vytvořený snímek je tvořen z MAFF objektů uložených v kořenové části archivu. Dle definované časové zarážky a výskytu objektů textového typu obsahujících skripty se původní obsah kořenové části dělí na jednotlivé snímky webové stránky. Snímek webové stránky, neboli *snapshot*, je část archivu transformující objekty v původním tvaru na otisk stránky v daném časovém intervalu.

První vytvořený snímek vždy obsahuje pouze základní hypertextové dokumenty a multimediální data, avšak žádný skript. Další vytvořené snímky vždy obsahují objekty z předchozího snímku a navíc obsahují všechny objekty i skripty načtené v intervalu před ukončením pomocí časové zarážky. Po ukončení intervalu je zkontrolováno, zdali se za časovou zarážkou nachází další objekty. V pozitivním případě se vytvoří další snímek, kde se ke stávajícím objektům přidají další objekty obsažené v následujícím intervalu.

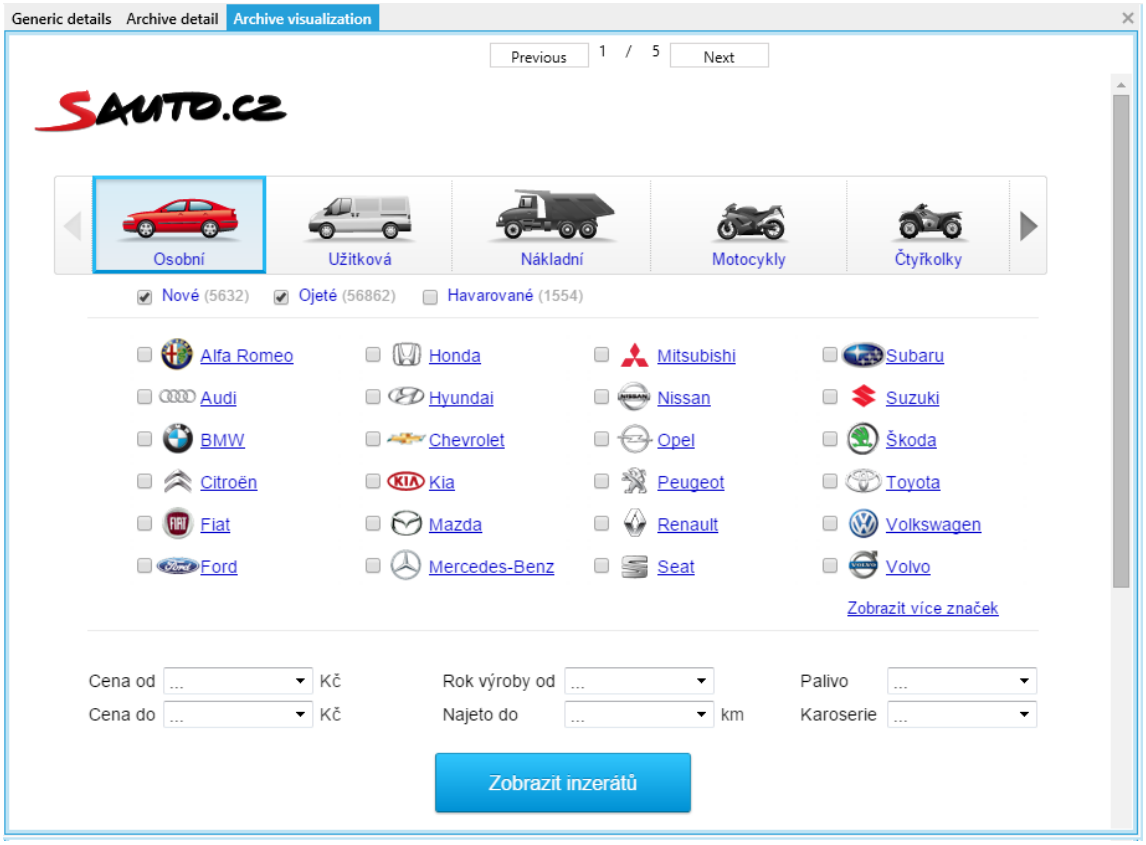
Po dokončení vytváření snímku se přechází na druhou část skládání, která řeší přepsání stávajících *referencí* objektů textového typu za *reference* odpovídající nynější struktuře archivu. Pro každou část archivu, tedy jak pro kořenovou část, tak pro část popsanou vytvořeným snímkem, jsou přepsány veškeré *reference* odkazující na původní zdrojový server. Původní *reference* jsou zaměněny *referencemi* odkazujícími na své potomky uložené v jednotlivé části archivu. Díky přepsání *referencí* se vytvoří stromová struktura stránky v rámci jedné části, a to při zachování větvení původní *struktury webové stránky*.

Posledním krokem při vytváření archivu je komprimace. Prvním krokem komprimace je vytvoření komprimovaného archivu pomocí třídy ***ZipArchive****(filestream, ziparchive mode)*. Po vytvoření komprimovaného archivu se začne vytvářet adresářová struktura MAFF archivu. Pro každou část archivu se vytvoří složka interpretující záložku MAFF archivu v prohlížeči. Do této složky se nahraje *stromová struktura stránky* uložena v dané části archivu. Na rozdíl od původního návrhu MAFF archivu popsaný v oddílu 2.4.2, kde každá záložka obsahovala samostatnou webovou stránku, jsou zde záložky využity pro zobrazení vytvořených snímků a původního obsahu zachycené webové stránky. To znamená, že první část, také známa jako *kořenová část archivu*, zobrazuje zachycenou webovou stránku v záložce stejně, jako by byla uložena technikou *browser saving system*. Oproti tomu jsou ostatní části archivu, tedy snímky webové stránky, uloženy technikou *snapshot*. Díky tomu lze dosáhnout lepší vizualizace uživatelských akcí. Po dokončení nahrávání všech části archivu je do komprimovaného souboru přidán soubor *index.rdf*, a to pro každou uloženou část. Tento soubor pak obsahuje metadata o jednotlivé uložené stránce. Následně se komprimovaný soubor uzavře a předá se jako výstup z MAFF modulu.

3.2.6 Vizualizace výstupních archivů

Druhá část MAFF modulu se zabývá vizualizaci a je popsaná abstraktním celkem *vizualizace archivu*. Tento abstraktní celek se dělí na dvě samostatné části vizualizace zobrazující požadované informace. První část se zaobírá zobrazením zachycené stránky v původním stavu. Tato část se bude nazývat *zobrazení stránky*. Druhá část se naproti tomu zabývá zobrazením objektů, které tyto archivy tvoří a také umožňuje spouštět každý objekt v jeho nativním formátu. Díky tomu zobrazuje souhrnné statistiky archivu a nazývá se *detail archivu*.

Prvním krokem při zobrazení stránky je nutná dekomprimace komprimovaného souboru. Důvodem dekomprimace je absence technologie pro přímé zobrazení archivu z komprimovaného souboru. Z toho důvodu je rovněž použita alternativní technologie *CefSharp* zmíněná v oddílu 3.1.3, která vyžaduje dekomprimovaný obsah archivu. Po dekomprimaci se zobrazí první část známá jako kořenová část archivu. Ve vizualizačním okně lze poté přecházet mezi jednotlivými částmi archivu a zaměřovat se na změny stránky pomocí uživatelské interakce.

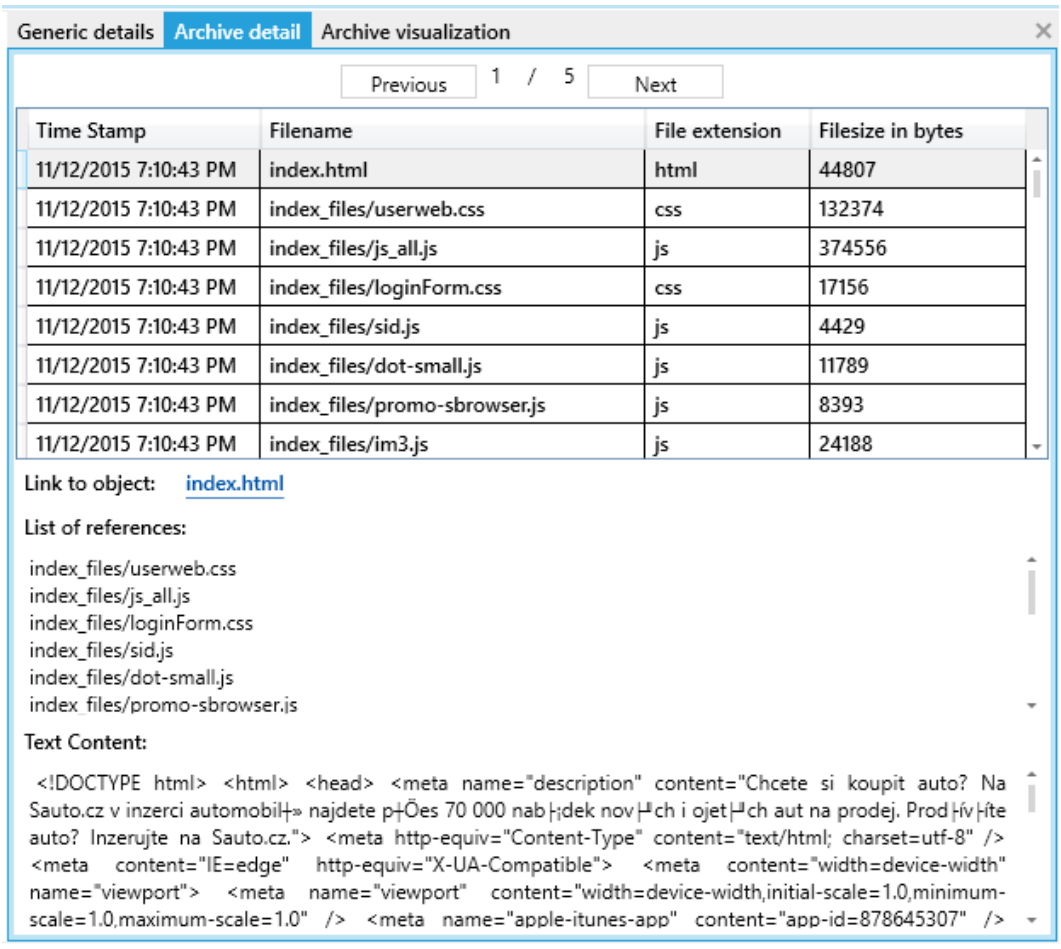


Obrázek 3.8: Vizualizace výsledného archivu

Na obrázku 3.8 je zrekonstruována stránka www.sauto.cz. Začátek webové stránky je mírně posunut kvůli interaktivním přepínačům mezi jednotlivými částmi. Kdykoliv lze přepnout na další část a věnovat se již provedeným změnám. V případě, že pro uživatele bude nutné vidět vedle vizualizačních změn i změny obsahu, může kdykoliv přepnout na *detail archivu*.

Uživateli se v detailu archivu zobrazí veškeré souhrnné statistiky všech těch MAFF objektů, které si může zobrazit v nativním formátu. Pro každý objekt se mu zobrazí čas zachycení, jméno, umístění, velikost obsahu a přípona souboru. Všechny tyto vlastnosti tvoří jednotlivé sloupce, na které lze aplikovat filtry umožňující lepší a rychlejší vyhledávání specifických informací. *Detail archivu* je zobrazen na obrázku 3.9. Vyjma uvedených vlastností lze u každého objektu vidět spustitelný odkaz tohoto objektu. Tento odkaz spustí každý soubor s určitou příponou v jeho výchozím programu. Například v případě spuštění HTML souboru se spustí výchozí prohlížeč, zatímco odkaz u obrázku spustí po kliknutí výchozí prohlížeč fotek a obrázků.

Další vlastnosti v *detailu archivu* se tykají objektů textového typu. Každý tento objekt dokáže zobrazit *reference* na své potomky. Díky tomu lze dohledat sestavení *stromové struktury stránky*. Poslední vlastností *detailu archivu* je zobrazení samotného obsahu objektu textového typu. V případě, kdy je označen soubor *index.html* lze vidět jeho obsah tvořený HTML značkami. Jak zobrazení *referencí* tak i zobrazení obsahu objektu lze ve vizualizaci dynamicky posouvat.



Obrázek 3.9: Detaily obsahu výsledného archivu

Kapitola 4

Testování a ověření výsledků

Kapitola se zaměřuje na otestování funkčnosti implementace modulu pomocí definovaných testů. Testy byly vytvořeny primárně za účelem objevení potencionálních nedostatků modulu a ověření stávájící implementace. Ověření výsledků jednotlivých testů bylo prostřednictvím referenčního *pluginu* do prohlížeče *Firefox*. Více informací o *pluginu* v podkapitole 2.4.

4.1 Charakteristika testů

Implementace modulu byla v průběhu vývoje podrobena sadám testů, které ověřují výslednou funkcionalitu. Každá sada testů se skládá z pevně definovaných vstupů obsahující zachycenou komunikaci. Zachycená komunikace je uložená v PCAP souborech. Jednotlivá sada testů obsahuje jeden a více vstupů pro větší možnost ověření funkcionality. Každý vstup je jedinečný a dělí se na vzorky obsahující zachycené webové stránky. Díky tomu může sada testů ověřovat vícenásobnou rekonstrukci z jednoho či více vstupu.

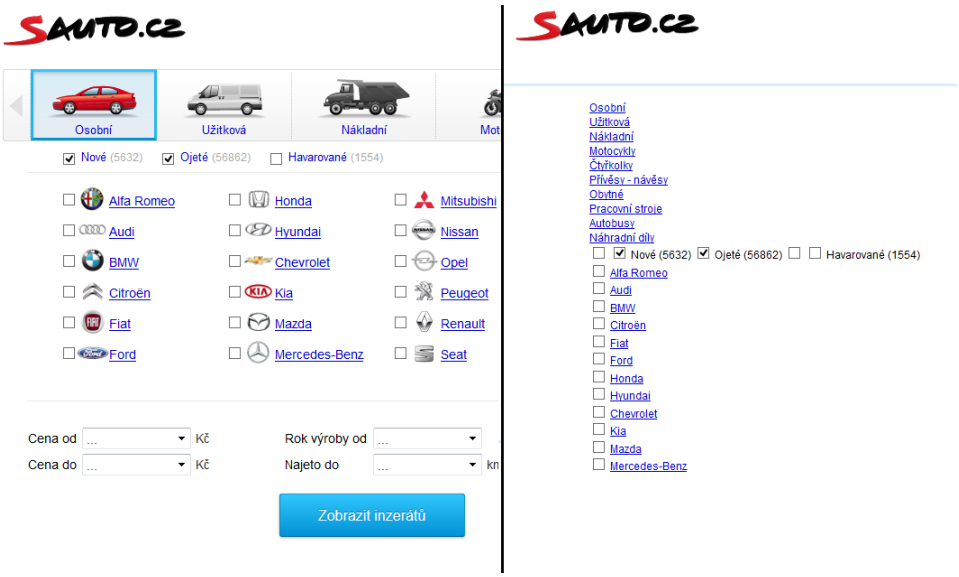
Testovací sady jsou dále rozděleny na dvě základní kategorie dle složitosti, počtu vstupů a jejich vzorků. Rozdíl mezi základními kategoriemi je v přístupu získaní vstupu a jejich vzorků. První kategorie testovacích sad byla zachytávána vždy po vyčištění vyrovnávací paměti prohlížeče. Důsledkem čištění paměti je přenos veškerého obsahu webové stránky skrze síťovou komunikaci při načítaní stránky. Druhá kategorie získávala vstupy bez vyčištění vyrovnávací paměti. To mělo za důsledek částečný přenos obsahu webové stránky, poněvadž již načtené objekty se znova nepřenesly.

Každá jednotlivá kategorie se dále člení na dvě podkategorie, které se dělí dle obsahu zachycených webových stránek. První podkategorie ověřuje funkčnost pomocí statických a polodynamických webových stránek. Statická webová stránka obsahuje prvky neměnné při jakékoliv uživatelské interakci. V praxi to znamenaná, že statická stránka neobsahuje žádné skripty. V době moderních webů se statické stránky příliš nepoužívají, a proto je první podkategorie doplněna o polodynamické webové stránky. Polodynamická stránka obsahuje dynamické prvky jako jsou reklamy nebo proměnlivé zobrazení. V dnešní době do kategorie polodynamických stránek patří většina webových stránek obsahující zpravodajství. Nejen, že se v době přístupu na webovou stránku mění dynamické prvky, uživatel může rovněž lehce ovlivnit vizualizaci stránky pomocí uživatelské interakce. Druhá podkategorie obsahuje plně dynamické webové stránky. Plně dynamická webová stránka obsahuje dynamické prvky, které jsou přímo závislé na uživatelské interakci. Do této kategorie patří sociální sítě, multimediální webové stránky a další weby pro rychlou výměnu informací.

4.2 Výsledky testů

Testování jednotlivých sad testů probíhalo po dvojicích ve dvou cyklech. V prvním cyklu se testovala první podkategorie z první a druhé základní kategorie. V druhém cyklu se testovala druhá podkategorie z první a druhé kategorie. Díky tomuto přístupu se zkombinovaly mezi sebou základní podkategorie pro lepší a efektivnější možnost ověření funkčnosti modulu.

Z odstavce výše uvedeného vyplývá, že první cyklus byl zaměřen na rozdíl obsahu statických a polo dynamických webových stránek při vyčištění nebo ponechání vyrovnávací paměti prohlížeče v rámci vytváření vstupu pro testovací sady. Po dokončení testů z prvního cyklu lze z porovnávání jednotlivých testovacích sad, dozvědět velmi rozdílné výstupy. Výsledek prvního cyklu je zobrazen na obrázku 4.1.



Obrázek 4.1: Výsledky testu v rámci prvního testovacího cyklu

Výsledky druhého cyklu, tedy testování plně dynamických webových stránek bylo tak rozdílné, že vizualizace nebyla potřeba. Kvůli absenci některých webových objektu v síťovém provozu při ponechání vyrovnávací paměti prohlížeče se vůbec nezobrazil obsah plně dynamické webové stránky.

Při vytváření souhrnu statistik jednotlivých testovacích sad je úspěšnost interpretovává v rámci každé podkategorie. První podkategorie v rámci kategorie při vyčištěné vyrovnávací paměti prohlížeče měla úspěšnost rekonstruovaných stránek kolem 80-100%. Druhá podkategorie při vyčištěné vyrovnávací paměti, tedy plně dynamické webové stránky, měla úspěšnost rekonstruovaných stránek kolem 30-80%. Rozptyl úspěšnosti je velký z důvodu výrazného ovlivnění měření každou plně dynamickou webovou stránkou. Při ponechané vyrovnávací paměti prohlížeče byla úspěšnost rekonstrukce webové stránky v rámci první podkategorie okolo 50-70%. U druhé podkategorie při ponechání vyrovnávací paměti byla úspěšnost rekonstrukce nejnižší a to v rozmezí 0-20%.

Kapitola 5

Závěr

Cílem této bakalářské práce byl návrh a realizace modulu určeného pro exportování obsahu webového provozu do MAFF, včetně vizualizace tohoto exportovaného obsahu. Modul byl implementován jako rozšiřující modul pro bezpečnostní nástroj *Netfox Detective*. Nejprve bylo potřeba se seznámit s principy přenosu webových objektů skrze Internet prostřednictvím aplikačního protokolu HTTP. Další pro danou problematiku podstatnou záležitostí bylo nastudování exportních, též archivačních, formátů pro uložení webových stránek a jejich objektů.

Zadání bakalářské práce dále uvádělo za podstatné seznámení s archivačním formátem MAFF, včetně jeho struktury, principů zobrazení a možností interpretace dat. V neposlední řadě bylo potřeba nastudovat bezpečnostní nástroj *Netfox Detective*, seznámit se s jeho exportními výstupy a začlenit modul do tohoto nástroje.

Modul vytvořený pro exportování obsahu webového provozu umožňuje získat a zobrazit webové stránky obsažené v zachycené komunikaci ve formě konverzací poskytnutých od bezpečnostního nástroje. Volitelné vlastnosti modulu umožňují základní změny ve vytváření archivů nesoucích webové stránky skrze konfiguraci pomocí nástroje. Struktura modulu byla navržena tak, aby dokázala interpretovat vizualizaci získané webové stránky přímo v nástroji.

Funkcionalita modulu byla úspěšně testována na základní množině vzorků zachycené komunikace potřebné pro stabilní výsledky. Při vývoji modulu byly vytvářeny rozdílové testy, které sloužily pro porovnávání výstupních archivů. V každém výstupním archivu byl porovnáván změněný obsah, tak aby se odhalili nedostatky navržených změn. Všechny rozdílové testy byly porovnávány s referenčním řešením MAFF pluginu prohlížeče. Při porovnávání byl zohledněn faktor rekonstrukce stránky ze zachycené komunikace nebo přímo z paměti prohlížeče využívajícího daný plugin. Po získaní testovacích výsledků byly určeny tři kategorie webových stránek a to podle míry rekonstrukce stránky. Tyto kategorie lze popsat jako statické, polodynamické a plně dynamické webové stránky. Statické a polodynamické webové stránky dosahují většinové míry úspěšností rekonstruované stránky. Zatímco plně dynamická webová stránka je hůře rekonstruovatelná a míra úspěšnosti rekonstrukce je nízká.

Rozšiřitelnost modulu je v budoucnu nezbytná pro zachování kvality rekonstruovaných stránek. Mezi největší důvody pro rozšíření modulu patří nově ustanovená verze standardu pro výměnu webových, potažmo hypertextových dokumentů, tedy HTTP verze 2[1]. Tato verze standardu totiž není zohledněna ve stávající struktuře implementovaného modulu. Další důvod pro rozšíření modulu je neustálý vývoj skriptů umožnující dynamické webové stránky. Pro zachování kvality by bylo vhodné rozšířit modul o komplexní analýzu skriptů a nejlépe dokázat jejich interpretaci přímo v modulu.

Potencionální využití modulu je spjato s bezpečnostním nástrojem *Netfox Detective*, který obohacuje o vizualizaci a rekonstrukci webových stránek. Využití tohoto nástroje lze nalézt v řadě případů. Může sloužit jako potencionální ochrana proti bezpečnostním hrozbám jako jsou podvodné webové stránky nebo nebezpečné reklamy.

[1] HTTP/2.0 - https://tools.ietf.org/html/rfc7540

Literatura

[1] *.NET (WPF and Windows Forms) bindings for the Chromium Embedded Framework*. [Online; navštíveno 11.4.2016].
URL https://github.com/cefsharp/CefSharp

[2] *SSL Handshake*. [Online; navštíveno 27.3.2016].
URL http://www.ibm.com/support/knowledgecenter/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660a.gif

[3] *The Mozdev*. Mozdev, [Online; navštíveno 4.4.2016].
URL http://www.mozdev.org

[4] Berners-Lee, T.: *RFC 1630: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web*. Informational, 1994, [Online; navštíveno 27.3.2016].
URL http://www.ietf.org/rfc/rfc1630.txt

[5] Berners-Lee, T.; Connolly, D.: *RFC 1866: Hypertext Markup Language*. Historic, 1995, [Online; navštíveno 1.4.2016].
URL https://tools.ietf.org/html/rfc1866

[6] Berners-Lee, T.; Fielding, R.: *RFC 1945: Hypertext Transfer Protocol – HTTP/1.0*. Informational, 1996, [Online; navštíveno 25.3.2016].
URL https://tools.ietf.org/html/rfc1945

[7] Consortium, W. W. W.: *Document Object Model (DOM)*. [Online; navštíveno 5.4.2016].
URL https://www.w3.org/TR/REC-DOM-Level-1/

[8] Consortium, W. W. W.: *Extensible Markup Language (XML)*. [Online; navštíveno 6.4.2016].
URL https://www.w3.org/XML/

[9] Corporation, M.: *Visual C# Language*. [Online; navštíveno 10.4.2016].
URL https://msdn.microsoft.com/en-us/library/aa287558(v=vs.71).aspx

[10] Dierks, T.; Rescorla, E.: *RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2*. Standards Track, 2008, [Online; navštíveno 27.3.2016].
URL https://www.ietf.org/rfc/rfc5246.txt

[11] Fielding, R.; Gettys, J.: *RFC 2616: Hypertext Transfer Protocol – HTTP/1.1*. Draft Standard, 1999, [Online; navštíveno 25.3.2016].
URL https://tools.ietf.org/html/rfc2616

[12] Freed, N.; Borenstein, N.: *RFC 2045-50: Multipurpose Internet Mail Extensions Part One-Five*. Draft Standard, 1996, [Online; navštíveno 2.4.2016].
URL https://tools.ietf.org/html/rfc2045

[13] Katzem, P.; Inc., P.: *.ZIP File Format Specification, V.: 6.3.4*. APPNOTE, 2014, [Online; navštíveno 5.4.2016].
URL http://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT

[14] Khare, R.; Lawrence, S.: *RFC 2817: Upgrading to TLS Within HTTP/1.1*. Proposed Standard, 2000, [Online; navštíveno 27.3.2016].
URL https://tools.ietf.org/html/rfc2817

[15] Ottley, C.; Amadini, P.: *About the MAFF file format*. Mozdev, 2004, [Online; navštíveno 4.4.2016].
URL http://maf.mozdev.org/maff-file-format.html

[16] Ottley, C.; Amadini, P.: *The MAFF specification*. Mozdev, 2004, [Online; navštíveno 4.4.2016].
URL http://maf.mozdev.org/maff-specification.html

[17] Palme, J.; Hopmann, A.: *RFC 2557: MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)*. Proposed Standard, 1999, [Online; navštíveno 1.4.2016].
URL https://tools.ietf.org/html/rfc2557

[18] Pluskal, J.: *Framework for captured network communication processing*. Diplomová práce, Faculty of Information Technology, Brno University of Technology, the Czech Republic, 2014.

[19] Wireshark.org: *Chapter Introduction*. [Online; navštíveno 28.3.2016].
URL https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html

[20] Wireshark.org: *Sample Captures*. [Online; navštíveno 28.3.2016].
URL https://wiki.wireshark.org/SampleCaptures

Přílohy

Seznam příloh

A Obsah CD **41**

Příloha A

Obsah CD

- Adresář *source* obsahuje zdrojové kódy modulu v C# jazyce.
- Adresář *doc* obsahuje zdrojové kódy textové části bakalářské práce pro LaTeX.
- Adresář *pcaps* obsahuje zachycené vstupní data pro testování.
- Adresář *archives* obsahuje výsledky testů ve formě výstupních archívů.
- Soubor *BP-VIT-JANECEK-2016.pdf* obsahuje textovou část bakalářské práce.