ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Podpora pro filtrování SSL/TLS komunikace v Privoxy
Student:	Václav Švec
Vedoucí:	Ing. Josef Kokeš
Studijní program:	Informatika
Studijní obor:	Informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Analyzujte způsoby, jak lze filtrovat HTTPS komunikaci, a bezpečnostní aspekty jednotlivých technik.
Seznamte se s problematikou proxy serverů ve vztahu k HTTP/HTTPS.
Navrhněte implementaci HTTPS filtru do programu Privoxy (www.privoxy.org) tak, aby využila maximum filtrovacích možností tohoto programu.
Naimplementujte základní verzi filtrování HTTPS.
Zhodnoťte dosažené výsledky a navrhněte možná budoucí vylepšení.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 27. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Podpora pro filtrování SSL/TLS komunikace v Privoxy

Václav Švec

Vedoucí práce: Ing. Josef Kokeš

9. května 2017

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce, panu Ing. Josefu Kokešovi, za jeho příkladnou ochotu a obětavost při vedení této práce. Dále děkuji své sestře Bc. Stanislavě Švecové za korekci textu práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. května 2017 .....

Odkaz na tuto práci

Švec, Václav. *Podpora pro filtrování SSL/TLS komunikace v Privoxy*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Práce se věnuje problematice proxy serverů ve vztahu k HTTP a HTTPS protokolům, a také popisuje způsoby filtrování HTTPS komunikace. Představeny jsou zde jednotlivé způsoby filtrování HTTPS komunikace, možnosti těchto filtrů a postupy proxy serveru při obsluze HTTP/HTTPS spojení.

Součástí práce je také návrh a implementace rozšíření programu Privoxy. Implementované rozšíření umožňuje filtrovat HTTPS komunikaci s využitím řady filtrovacích možností původního programu. Dosažené výsledky jsou zhodnoceny a doplněny o další možná vylepšení.

Klíčová slova proxy server, filtrování, Privoxy, SSL, TLS, HTTPS

Abstract

The thesis deals with the topic of proxy servers in relation to HTTP and HTTPS protocols and describes the ways of filtering HTTPS communication. Different ways of HTTPS filtering are presented, as well as the features of these filters and the proxy server procedures for controlling of HTTP/HTTPS connections.

The second part of this thesis implements a SSL extension for Privoxy. This extension allows filtering HTTPS communication using variety of filtering features of the original program. Achieved results are evaluated and other possible improvements are suggested.

Keywords proxy server, filtering, Privoxy, SSL, TLS, HTTPS

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době nás internet spojuje s celým světem téměř odkudkoliv a kdykoliv. Přenášeny jsou dříve nepředstavitelné objemy dat relativně vysokými rychlostmi na veliké vzdálenosti. Takové datové přenosy však přináší i řadu požadavků na jejich kontrolu a případnou úpravu či blokování. Typickým příkladem může být velké množství reklamních sdělení na webových stránkách. Uživatelé často hledají řešení pro odstranění těchto reklam při zachování podstatných částí webové stránky.

Jednou z překážek filtrování síťové komunikace je šifrování přenášených dat pomocí protokolu HTTPS. To znemožňuje filtračnímu softwaru získat z přenášených dat potřebné informace a ten nemůže uplatnit své filtry. Postupné nasazování protokolu HTTPS na velkou část webových serverů tak způsobuje, že aplikování filtrů v proxy serveru Privoxy a podobných programech je pro velkou část webových stránek nemožné.

Cílem této bakalářské práce je návrh a implementace rozšíření programu Privoxy, které umožní filtrovat HTTPS komunikaci s využitím filtrovacích možností tohoto proxy serveru. Návrh a implementace vychází z teoretických základů získaných během analýzy způsobů filtrování HTTPS komunikace a analýzy problematiky proxy serverů ve vztahu k protokolům HTTP a HTTPS. Tyto analýzy jsou součástí práce. Dále práce obsahuje zhodnocení dosažených výsledků, doplněné o možná budoucí vylepšení.

Téma jsem si zvolil především pro jeho praktické využití. Implementované rozšíření umožňuje uživatelům využívat filtrování komunikace z programu Privoxy nejen pro nešifrované webové stránky, ale i pro šifrované. Díky tomu jsou značně rozšířeny stávající možnosti programu Privoxy. Navíc je tato funkcionalita poptávána nemalou skupinou uživatelů. Dalším důvodem pro zvolení tohoto tématu je můj zájem o síťové technologie, které se v této práci spojují s programováním a bezpečností.

Kapitola 1

Způsoby filtrování komunikace

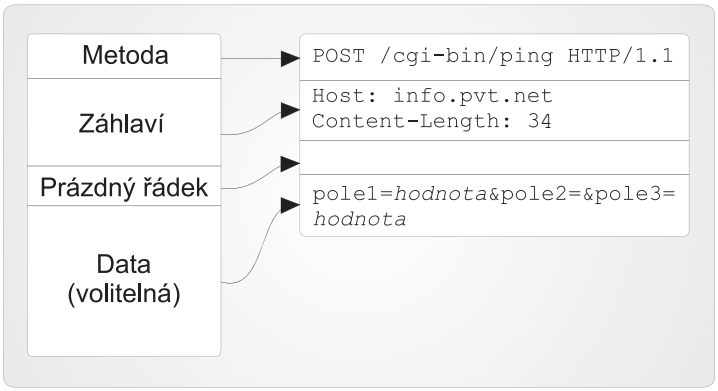
Během zpracování požadavku klienta na stažení určité webové stránky je mnoho vhodných míst k přerušení komunikace a aplikování filtrů. Každá z možných variant s sebou přináší různé výhody a nevýhody vyplývající právě z umístění filtru na trase přenášených dat. Tato kapitola popisuje základní principy protokolu HTTP, který je používán pro přenos dat mezi klientem a webovým serverem. Následně jsou popsány jednotlivé postupy, kterými lze přenášenou komunikaci filtrovat. Tyto metody mohou být zvoleny například v závislosti na možnostech uživatele či organizace, která chce filtrování komunikace nasadit, případně podle požadavků na vlastnosti použitých filtrů.

1.1 Protokol HTTP

Po zapsání URL adresy do okna prohlížeče dojde k přeložení doménového jména na IP adresu. Poté je navázáno TCP spojení mezi klientem a webovým serverem a následně již klient odesílá HTTP dotaz, na který ve stejném spojení odpovídá webový server HTTP odpovědí. Webové stránky se obvykle skládají z několika objektů; pro každý objekt je nutný jeden HTTP dotaz. Z odpovědí již prohlížeč sestaví podobu webové stránky.[8]

Komunikace mezi klientem a serverem je tedy tvořena posloupností dotazů a odpovědí, avšak každý požadavek skládající se z dotazu a odpovědi je samostatný. To znamená, že veškeré informace potřebné k reakci na dotaz musí být obsaženy v tomto jediném dotazu a nelze je získat z předchozí komunikace. Díky tomuto principu je HTTP označován jako bezstavový protokol. Kvůli tomu musí být veškeré stavové informace o připojení klienta spravovány jinou metodou. Nejčastěji je k tomuto účelu používána metoda cookies souborů.[8, 18]

Existuje několik verzí protokolu HTTP. Novější verze přináší mnohá vylepšení a spolu s nimi i nové parametry. Například od HTTP/1.1 již implicitně není pro každý dotaz navazováno nové TCP spojení, navíc je možné odeslat více dotazů bez čekání na vyřízení předchozího.[8]



Obrázek 1.1: Struktura HTTP dotazu[8]

1.1.1 HTTP dotaz

Struktura HTTP dotazu se může lišit v závislosti na verzi protokolu. Pro ukázku je zde popsán HTTP dotaz ve verzi 1.1. První řádka obsahuje tzv. metodu, URI požadovaného objektu a verzi HTTP protokolu. Ve verzi HTTP/1.1 jsou podporovány metody GET, POST, HEAD, CONNECT, PUT, DELETE, TRACE a OPTIONS. Dále následuje záhlaví, které se skládá z jednotlivých hlaviček. Hlavička obsahuje klíčové slovo zakončené dvojtečkou a mezerou, za kterou mohou být zapsány parametry hlavičky. Parametry pak musí být ukončeny koncem řádku CRLF. Jediná povinná hlavička je hlavička Host. Záhlaví je zakončené prázdnou řádkou. Dále už mohou pokračovat přenášená data. Ukázková struktura HTTP dotazu je zobrazena na obrázku 1.1.[8]

1.1.2 HTTP odpověď

HTTP odpověď začíná stavovým řádkem obsahujícím verzi HTTP, třímístný stavový kód a poznámku. Stavový kód určuje, jestli operace proběhla úspěšně či nikoliv, případně k jaké došlo chybě. Poznámka pak jen doplňuje stavový kód. Po stavovém řádku následuje opět záhlaví s jednotlivými hlavičkami. Za záhlavím je prázdný řádek a následně přenášená data. V tabulce 1.1 jsou zobrazeny jednotlivé druhy stavových kódů, které se mohou v odpovědi vyskytnout. Druh stavového kódu je určený první číslicí z trojice.[8]

1.2 Paketové filtry

Základní možností filtrování komunikace je využití paketových filtrů. Paket je datový balíček s omezenou velikostí, který přenáší data po síti. Filtrování paketů může být prováděno na linkové nebo síťové vrstvě modelu ISO/OSI. Pro linkovou vrstvu k tomuto účelu slouží switche (přepínače), pro filtrování

Tabulka 1.1: Druhy stavových kódů HTTP protokolu[8]

1xx	Informativní odpověď, zpracování pokračuje dále.
2xx	Akce proběhla úspěšně.
3xx	Přesměrování, tj. další akce se bude týkat jiného URI.
4xx	Chyba klienta.
5xx	Chyba serveru.

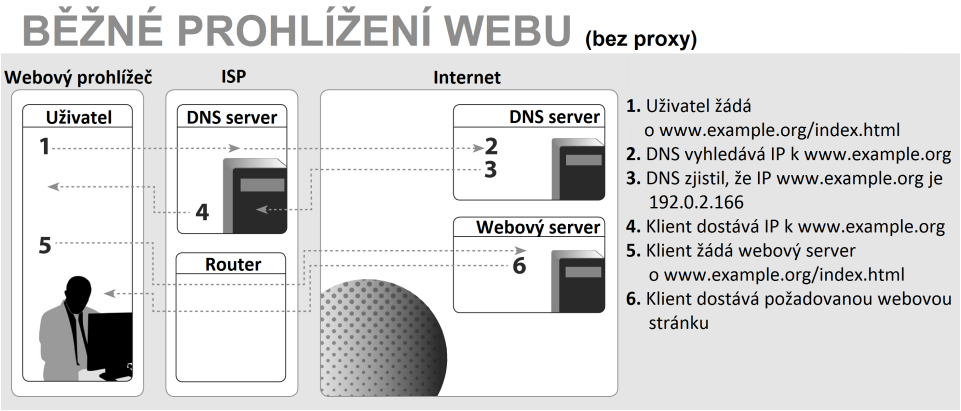
na síťové vrstvě lze použít router (směrovač), ale stejně tak může být použit i firewall pro obě vrstvy. Switche a routery slouží k přenosu paketů od klienta k serveru a zase zpět. Pro aplikování filtru na určité vrstvě je nezbytné, aby dané zařízení dokázalo přečíst záhlaví protokolů této vrstvy. Právě v těchto záhlavích jsou zapsány informace, podle kterých filtry rozhodují. Filtry mají seznam MAC adres (respektive IP adres), které mají být propuštěny nebo zablokovány. Navíc se zde rozlišují adresy odesilatele a příjemce datového paketu. Pakety určené k blokování nejsou přeposlány dále.[8]

Použitím paketových filtrů může uživatel blokovat komunikaci s určitými MAC/IP adresami, nebo mezi určitými MAC/IP adresami, a to v libovolném směru. Tato metoda však sama o sobě nepřináší mnoho možností, zejména neumožňuje sledovat obsah přenášených paketů a filtrovat podle něj, a tak bývá používána v kombinaci s dalšími, které jsou uvedeny níže. Nelze také filtrovat na základě parametrů vyšších vrstev modelu ISO/OSI. Výhodou je naopak jednoduché zavedení a konfigurace a vysoký výkon.

1.3 Manipulace DNS

Určitého filtrování komunikace může být dosaženo také pomocí DNS protokolu. Tento způsob filtrování komunikace vychází ze samotného principu stahování webových stránek.

Pro vyřízení dotazu jsou používány protokoly ze sady protokolů TCP/IP. Ty přenášejí data mezi klientem a serverem v podobě paketů. Pakety jsou v rámci internetu řízeny zařízeními zvanými router. Router je připojen k několika linkám a o každém příchozím paketu rozhoduje, na kterou linku bude přeposlán. Toto rozhodnutí je ovlivněno cílovou IP adresou, která je uložena v každém paketu. Routery tak směrují přenášená data až k cílovému serveru, na který klient zaslal svůj požadavek.[6] Jelikož však existuje velké množství webových stránek a uživatelé by si jejich IP adresy obtížně pamatovali, umožňuje DNS (Domain Name System) používat doménová jména, která spojuje s IP adresami. Uživatel místo IP adresy zadává doménové jméno, které následně bude přeloženo na IP adresu. To mu umožní speciální servery, tzv. DNS resolvery, které vyhledají doménové jméno a IP adresu ve své databázi, případně přepošlou dotaz na jiný DNS resolver. Tyto servery jsou obvykle



Obrázek 1.2: Komunikace mezi klientem a serverem (Upraveno podle [6])

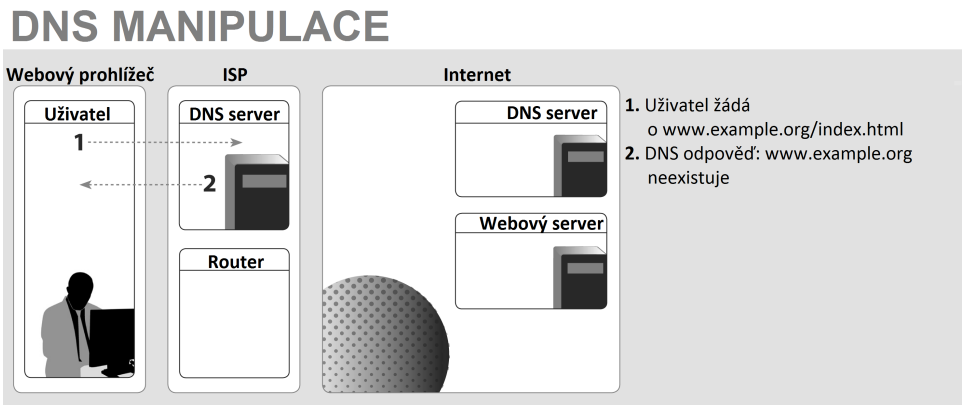
spravovány poskytovateli internetového připojení[1].[3] Po získání IP adresy cílového webového serveru může klient odeslat svůj dotaz a cílovou IP adresu uvést v jednotlivých paketech. Díky tomu routery správně určí cestu ke konkrétnímu webovému serveru, a ten pak odešle odpověď na původní adresu odesilatele. Popsaný princip komunikace je zobrazen na obrázku 1.2.[6]

Díky tomu, že většina uživatelů i odkazů v rámci webových stránek využívá doménová jména místo IP adres, může být manipulací s DNS serverem dosaženo blokování zakázaných stránek (ať už se jedná o stránky šířící reklamu, nebezpečný obsah nebo stránky blokované z jiného důvodu). Pokud klient požádá o překlad zakázaného doménového jména na IP adresu, DNS server vrátí chybovou zprávu. Bez IP adresy pak zařízení uživatele nemůže kontaktovat webový server a místo požadovaného obsahu zobrazí chybové hlášení. Na obrázku 1.3 je tento postup zobrazen.[6]

Manipulace DNS umožňuje kromě blokování určitých doménových jmen také přesměrovat doménová jména na jiný webový server. Pokud klient požádá o překlad vybraného doménového jména na IP adresu, DNS server může vrátit IP adresu jiného webového serveru (falešnou IP adresu). Klient pak odešle své požadavky jinému webovému serveru. Pouze při přesměrování HTTPS protokolu je nutné, aby webový server odeslal důvěryhodný certifikát, jinak je toto přesměrování odhaleno webovým prohlížečem.

Tento způsob filtrování tedy umožňuje blokování webových stránek nebo přesměrování na jiný webový server. Přináší však také řadu nevýhod. DNS filtr přímo nezabraňuje přístupu k webové stránce, ale pouze zabraňuje přístupu s použitím doménového jména. Problematická webová stránka je tak nadále přístupná při použití jiných DNS serverů či s použitím IP adresy místo doménového jména. Obcházení této metody filtrování je tedy velmi jednoduché.

[1]poskytovatel internetového připojení (ISP) – společnost poskytující uživatelům přístup k internetu



Obrázek 1.3: Blokování webové stránky pomocí DNS (Upraveno podle [6])

Obranou proti přesměrování na jiný webový server pak může být například protokol DNSSEC, který odhalí záměnu IP adresy webového serveru.[5] Další nevýhoda této filtrovací metody spočívá v nízké rozlišovací schopnosti. Jelikož klient nezíská cílovou IP adresu k danému doménovému jménu, dochází nutně k blokování veškerého obsahu této domény.

Naopak mezi výhody tohoto způsobu filtrování patří například jednoduché zavedení. Často se může jednat pouze o úpravu již existujících zařízení. Další výhodou jsou velmi nízké nároky na výpočetní výkon či efekt na velké množství uživatelů. Navíc jsou blokovány stránky dostupné přes HTTP i HTTPS protokol.

DNS filtrování využívají například státy jako Vietnam, Jižní Korea nebo Indie. Pro zvýšení efektivity je v těchto zemích metoda DNS filtrování používána spolu s dalšími blokovacími technikami.[6]

1.4 Firewall

Firewally kontrolují komunikaci na hranicích privátních sítí s externími sítěmi. Pokud paket procházející firewallem splňuje nastavená pravidla, je propuštěn dále. V opačném případě je paket zablokován. Firewally tak kontrolují, schvalují a zamítají jednotlivé pokusy o připojení mezi interní sítí a externími sítěmi a chrání tak uživatele uvnitř sítě před hrozbami zvenčí. Díky firewallům lze zabezpečit celou síť z jednoho místa a vnitřní zařízení díky tomu již nemusí provádět činnost firewallů a spoří svůj výkon. Použitím firewallu jsou soustředěny veškeré externí služby do jediného zařízení. Firewall však také vytváří úzká hrdla mezi interními a externími sítěmi, neboť veškerá komunikace mezi těmito sítěmi musí projít jediným zařízením. V rámci zařízení, které provozuje firewall, může být spuštěno hned několik služeb současně. Těmi základními jsou:[19]

- Filtrování paketů

  Odmítání paketů TCP/IP, které nevyhovují nastavení filtrů.

- Překládání síťových adres (NAT)

  Překládá IP adresy interních počítačů na adresy externí. Díky tomu jsou interní adresy skryty před monitorováním zvenčí. Navíc jsou tak šetřeny veřejné IP adresy.

- Služba proxy

  Tato metoda bude v rámci této práce popsána samostatně v následující části.

Firewall může být speciální zařízení, ale může pracovat i na serveru nebo v rámci jiného síťového prvku, například routeru. V rámci firewallu pak nemusí být další služby použity. Platí ovšem, že pokud jsou všechny služy v rámci jednoho zařízení, je toto řešení zpravidla bezpečnější.[19] V této části práce ovšem budeme předpokládat firewall bez služby proxy, abychom oddělili jednotlivé možnosti těchto metod. Jedinou metodou filtrování komunikace na firewallu tak zůstává paketový filtr.

1.4.1 Bezstavový paketový filtr

Původně byly firewally pouze jednoduché paketové filtry a filtrování paketů zůstalo dodnes hlavní funkcí firewallů. Paketový filtr porovnává protokoly síťové a transportní vrstvy (například TCP a IP protokoly) s databází pravidel, následně propouští pouze pakety splňující daná kritéria. Pro toto porovnání jsou nejčastěji použité parametry: IP adresa, port, typ protokolu, číslo fragmentu, či informace o přímém směrování. Ve filtrech jsou obvykle nastavena následující pravidla:[19]

- Blokování pokusů o připojení zvenčí
- Blokování paketů TCP na určitých portech, případně blokování paketů TCP na určitých portech, pokud směřují na danou IP adresu
- Omezení příchozích přístupů na vybrané IP adresy

Díky možnosti filtrovat určité porty lze paketovým filtrem blokovat HTTPS protokol blokováním portu 443. Toto blokování však lze obejít používáním jiných portů pro tento protokol.

1.4.2 Stavový paketový filtr

Bezstavový způsob filtrování lze snadno napadnout, neboť některé pakety nemusí obsahovat veškeré informace nutné pro správné rozhodnutí filtru. Příkladem může být fragmentace, kdy jsou informace o portu obsaženy pouze

v prvním fragmentu a všechny ostatní už tuto informaci neobsahují. Filtr pak blokuje pouze první fragment a ostatní mohou projít. Další problém spočívá v tom, že většina bezstavových filtrů povoluje veškeré porty nad 1024. Tyto porty jsou využívány pro zpětné připojení serveru na soket, a jelikož bezstavový filtr nedokáže určit, které spojení bylo vyžádáno z interní sítě a které ne, musí povolit všechna příchozí spojení.[19]

Novější stavové paketové filtry (někdy označované jako dynamické paketové filtry) navíc oproti bezstavovým filtrům logují data o odchozích paketech z interní sítě, a příchozí pakety jsou pak povoleny, pouze pokud se jedná o odpověď na již vyžádané spojení. Prakticky se tedy udržuje stav jednotlivých spojení. Díky tomu se lze bránit proti napadením pomocí fragmentace a zároveň není nutné mít povolené veškeré porty nad 1024. Díky informaci o stavu spojení se otevírají porty pouze pro očekávaná spojení. Tento typ paketových filtrů je výrazně bezpečnější a dnes také nejběžnější.[19, 4]

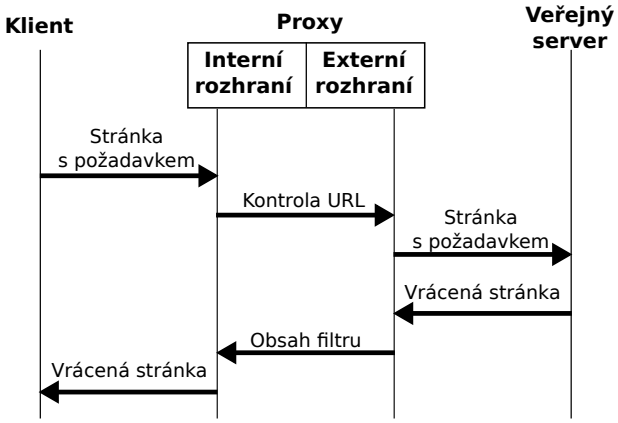
1.4.3 Kontrola protokolů (DPI – Deep Packet Inspection)

Novější aplikace umožňují pro svoji funkci využívat libovolný port, některé dokonce přenáší data pomocí protokolů určených k úplně jiným účelům. Jako příklad můžeme uvést aplikaci Skype. To způsobuje, že tyto aplikace nelze jednoduše omezit obyčejným paketovým filtrem. Pro získání potřebných informací je nutné získat informace od čtvrté až sedmé vrstvy modelu ISO/OSI. Firewally tak v přenášených datech vyhledávají typické chování pro jednotlivé protokoly a aplikace. Z tohoto důvodu musí být DPI pravidelně aktualizováno. Tato technologie tedy umožňuje firewallům filtrovat pakety aplikačních protokolů.[1]

1.5 Proxy

V knize [19] je princip proxy popsán následovně: „Proxy fungují tak, že naslouchají požadavkům o služby od interních klientů a pak je předávají na externí síť jako kdyby byl klientem – původcem samotný server proxy. Jakmile obdrží proxy od veřejného serveru odpověď, vrátí tuto odpověď původnímu internímu klientskému počítači, jako kdyby byl sám původním veřejným serverem.“ Tento princip je zobrazen na obrázku 1.4. Proxy server se tedy chová jako jakýsi prostředník mezi klientem a serverem.

Původně proxy sloužily především pro ukládání často navštěvovaných webových stránek do vyrovnávací paměti. Tím byl minimalizován počet připojení k internetu a opětovné stahování stránek. To bylo velmi výhodné především s dřívějším pomalým připojením. S postupným nahrazováním statických webových stránek dynamickými však využití této funkce značně kleslo. Postupně se ale našel nový potenciál proxy serverů, který vyplývá z opětovného generování požadavků klienta a z umístění proxy mezi klientem a serverem. Umožňují například skrýt všechny uživatele v interní síti za jediné zařízení, blokovat určité



Obrázek 1.4: Princip proxy (Překresleno podle [19])

URL adresy, filtrovat komunikaci a blokovat ji podle obsahu. Díky tomu dnes slouží především pro účely bezpečnosti.[19]

1.5.1 Aplikační proxy versus obvodové brány

Proxy jsou nejčastěji kombinovány se službou HTTP, protože právě pro ni byly původně vytvořeny. Dnes se ale využívají i pro většinu jiných internetových služeb. Pro každý protokol je však nutné rozšířit proxy o příslušný modul. Jednotlivé moduly většinou obsluhují pouze jeden konkrétní protokol, jako například HTTP, FTP či Telnet. Zároveň s protokoly je pak nutné aktualizovat i příslušné moduly proxy. Takové proxy servery, které rozumí struktuře přenášeného protokolu, jsou nazývány aplikační proxy (application-level proxy).[19, 22]

Mnoho protokolů však vlastní některé společnosti nebo se příliš nepoužívají, a tak pro ně neexistuje žádný proxy modul. Pro tyto protokoly je pak nutné použít všeobecné proxy servery nazývané obvodové brány (circuit level gateways). Ty generují přijaté pakety znovu, avšak nedokáží interpretovat aplikační protokol. Proxy tak vytváří okruh mezi klientem a serverem, ve kterém pouze přeposílá pakety. Tento typ proxy často používá protokol SOCKS, který nabízí i další funkce, jako například autentizaci. Oproti klasickým paketovým filtrům má řešení pomocí obvodových bran tu výhodu, že nově vygenerované pakety nemohou obsahovat případné chyby v hlavičkách paketů. Takové chyby nemusí být firewally schopné odhalit. Další výhodou těchto proxy je také možnost nasazení pro velké množství různých protokolů bez nutnosti spravovat jednotlivé moduly. Nicméně ne každý protokol může být jednoduše obsluhován obvodovými bránami. Nevýhodou tohoto řešení pak je jen velmi nízká kontrola nad přenášenými daty v porovnání s aplikačními proxy. [19, 22]

1.5.2 Generické versus dedikované proxy

Dedikované proxy servery jsou takové, které pracují pouze s jedním aplikačním protokolem. Většina aplikačních proxy serverů jsou zároveň dedikované. Obsluhují tedy pouze jeden konkrétní protokol, například FTP. Na druhé straně generické proxy servery obsluhují několik různých protokolů zároveň. Většina obvodových bran jsou tudíž generické proxy servery. Z tohoto důvodu tyto pojmy často nebývají rozlišovány.[22]

1.5.3 Explicitní versus transparentní proxy

Explicitní proxy servery jsou takové servery, pro které je nutné nastavit na straně klienta odpovídající konfiguraci. Je tedy nezbytné pro každou službu, která má být používána prostřednictvím proxy, nastavit v příslušné aplikaci IP adresu a port, na kterém proxy server obslouží veškeré požadavky této služby. Některé aplikace usnadňují konfiguraci díky automatické detekci nastavení proxy na síti. Problém ale může nastat, pokud aplikace klienta nepodporuje spolupráci s proxy. Tento problém řeší tzv. transparentní proxy. Moderní firewally umožňují změnit směrování odchozího provozu na určitý počítač v síti. Toho lze využít k přesměrování určitých požadavků na konkrétní proxy server pro danou službu. Zároveň není potřeba, aby klient měnil jakákoliv nastavení, neboť firewall může směrovat odpovědi od proxy serveru zpět ke klientovi. Díky tomuto postupu odpadá veškeré nastavování na straně klienta.[19]

1.5.4 Reverzní proxy

Proxy servery lze také využít pro rozložení zátěže mezi několik webových serverů. Nové webové stránky často kladou vyšší nároky na výkon webových serverů a snižují tak maximální počet klientů, které může jeden server obsloužit. Díky reverznímu proxy serveru na straně cílových serverů lze připojit velké množství klientů na jednu cílovou IP adresu a proxy server pak rozloží požadavky na několik serverů za touto adresou podle jejich aktuálního vytížení. Protože proxy server pouze přeposílá požadavky, zvládne obsloužit větší množství klientů oproti jednomu samostatnému webovému serveru, který musí přijaté požadavky klientů zpracovat.[19]

1.6 Webový prohlížeč

Proces načtení webové stránky se z pohledu prohlížeče skládá ze tří kroků: Zažádání serveru o obsah webové stránky, přijmutí tohoto obsahu a následné zobrazení webové stránky.[20] Skutečnost, že k filtrování a blokování obsahu může docházet během těchto tří kroků přímo v prohlížeči, zcela odstraňuje problémy s šifrováním komunikace pomocí protokolu HTTPS. Jelikož prohlížeč sám iniciuje šifrované spojení, jsou veškerá data, která přijdou v prohlížeči

ke zpracování a zobrazení, již dešifrována. Zároveň lze na úrovni prohlížeče nejen filtrovat obsah komunikace, ale i ovlivňovat samotnou komunikaci se serverem. Většina starších webových prohlížečů během zobrazování webové stránky pouze vykonávala instrukce načtené z přijatých souborů a uživatel tento postup nemohl ovlivňovat. Dnešní prohlížeče však nabízejí hned dvě možnosti, jak tento postup řídit:[20, 10]

1. Blokování obsahu pomocí konfigurace prohlížeče
2. Blokování obsahu pomocí rozšíření pro prohlížeč

1.6.1 Blokování pomocí konfigurace prohlížeče

Nejjednodušším způsobem blokování nechtěného obsahu webových stránek je využití již existujících funkcí prohlížeče. Některé z nich pracují na základě heuristik a jiné spoléhají na zpětnou vazbu od uživatele. Jako příklad jsou zde uvedeny funkce popsané na blogu společnosti Microsoft [10], které nabízí prohlížeč Internet Explorer:

- Security Zones (Zóny zabezpečení)

  Tato funkcionalita umožňuje uživatelům přiřazovat pro jednotlivé webové stránky tzv. zóny s určitými privilegii. Těmito privilegii je pak omezen veškerý přijatý obsah. Webové stránky v omezené zóně pak například mohou spouštět skripty pouze s velmi nízkými oprávněními, nemohou využívat cookies, spouštět skripty třetích stran, načíst ActiveX prvky nebo stahovat soubory.

  Zóny zabezpečení však mají také řadu nevýhod. Například i přes omezení práv je blokovaný obsah webu stále stahován, ačkoliv poté není použit. Obsah stránek je omezen, ale nemusí být nutně blokován. Například obrázky z omezené zóny budou i přes omezení zobrazeny. Touto funkcionalitou také nelze zcela blokovat veškerý obsah webové stránky a s rostoucím počtem stránek dochází ke zpomalení prohlížeče.

- Per-Site ActiveX

  Pokud chce uživatel spravovat Flash, Silverlight, Javu nebo jiné doplňky v Internet Exploreru, je Per-Site ActiveX konfigurace tím správným nástrojem. Umožňuje totiž určit, které webové stránky mohou využívat tyto doplňky. Ve výchozím nastavení jsou veškeré doplňky pro všechny webové stránky povoleny, ale uživatel může sestavit vlastní seznam povolených stránek. Toto řešení má z hlediska uživatelů nedostatky v podobě obtížné konfigurace a velmi častých upozornění.

- InPrivate Filtering

  InPrivate Filtering umožnuje odhalit a blokovat obsah webových stránek načítaný od třetích stran. To umožňuje například blokovat reklamy,

které často bývají stahovány z webových serverů určených pouze pro reklamu. Při odhalování takového obsahu je využito buďto heuristiky, nebo rozhodnutí uživatele. Uživatel může sám označit určitý obsah k blokování. Prohlížeč pak už nebude provádět dotazy na tuto URL, pokud se bude jednat o obsah třetích stran v rámci webové stránky. Zároveň je možné importovat seznam blokovaných zdrojů. Je zde také možnost automaticky blokovat zdroje, které již byly odkazovány z určitého množství nezávislých webových stránek.

Hlavní nevýhody tohoto přístupu jsou nechtěné blokování určitého obsahu a problémy s blokováním sdílených knihoven. Některé populární knihovny jako JQuery jsou sdíleny velkými společnostmi a používány řadou webových stránek. To je výhodné z hlediska výkonu, ale zároveň to může vést k blokování jako nechtěný obsah od třetích stran.

- Cookie Controls

  Pomocí Cookie Controls lze omezovat použití cookies pro jednotlivé webové stránky. Umožňuje například blokovat veškeré cookies nebo blokovat cookies třetích stran. Cookies jsou rovněž spravovány pomocí zón. Ve výchozím nastavení jsou veškeré cookies povoleny.

- InPrivate Browsing (Anonymní režim)

  Tento režim lze použít, pokud si uživatel nepřeje ukládat určité informace o prohlížení webových stránek. Jedná se například o cookies či historii procházení.

1.6.2 Blokování pomocí rozšíření pro prohlížeč (tzv. extensions)

V první řadě je potřeba rozlišit rozšíření prohlížeče od pluginů, i když některé publikace jako například [20] tyto pojmy skriktně neodlišují. Pluginy jsou programy, které webovým prohlížečům umožňují zobrazit či zpracovat některé formáty souborů. Příklady takových pluginů jsou Adobe Reader pro PDF soubory nebo Adobe Flash Player pro videa. Oproti tomu rozšíření prohlížečům přidávají nové funkcionality. Jako příklad můžeme uvést AdBlock k odstranění reklam nebo rozšíření k upozornění na gramatické chyby v textu.[14] Pro účely filtrování lze využít právě možnosti rozšíření prohlížeče.

Na rozdíl od prohlížeče může jeho rozšíření implementovat prakticky kdokoliv. Tato rozšíření mohou doplňovat již existující prostředky prohlížeče o další funkce, případně nabídnout uživatelům jednoduší obsluhu a jiné výhody. Rozšíření lze rozdělit do dvou kategorií podle způsobu, jakým je prováděno blokování či filtrování obsahu. Jedná se buďto o automatické, nebo manuální blokování.[10]

1. Automatické blokování

Automatické blokování zasahuje do části prohlížeče, která je zodpovědná za stahování jednotlivých součástí webových stránek. Jsou zde zkoumány požadavky na server, a pokud žádají o blokovaný obsah, dojde ke zrušení takových požadavků ještě před jejich odesláním. Blokovaný obsah tak není nutné stahovat. Jako doplněk lze použít i zkoumání obsahu již stažených souborů. Pokud obsah souhlasí s definovaným vzorem, může být tento obsah odstraněn nebo pozměněn. Jako příklady automatického blokování můžeme uvést doplňky jako Simple Ad-Block, IE7Pro, Adblock Pro, AdBlock Plus a další.[10]

2. Manuální blokování

   Na rozdíl od automatického blokování umožňuje manuální blokování odstranit nechtěný obsah až po jeho stažení od serveru. Tento způsob blokování je jednoduší na implementaci, ale obvykle poskytuje méně možností. Často tato rozšíření představují pouze rozhraní pro konfiguraci již existujících řešení v prohlížeči. Manuální blokování bývá díky jednoduchosti rychlejší a spolehlivější. Na druhou stranu je nutné stažení celého obsahu webové stránky, čímž dochází ke zdržení.[10]

Kapitola 2

Filtrování komunikace

Filtrování komunikace rozšiřuje možnosti uživatelů nebo síťových administrátorů o řadu funkcí. Filtrování umožňuje například odstranit z webových stránek nechtěný nebo dokonce nebezpečný obsah, případně tento obsah nahradit jiným. Existuje řada možností, jak filtrování komunikace využít pro potřeby uživatelů, a právě na tyto možnosti se zaměřuje tato kapitola. Následující popis vychází zejména z dokumentace k proxy serveru Privoxy[15] se zahrnutím dalších zdrojů. Pro vyšší přehlednost jsou jednotlivé možnosti rozděleny do několika částí podle účelu.

Pro nastavení filtrů Privoxy slouží dva hlavní soubory. První z nich definuje jednotlivé filtry a druhý přiřazuje (nebo odebírá) tyto filtry k určité množině webových stránek či souborů na těchto stránkách, případně k určitým portům. Pro jednu webovou stránku pak lze jednotlivé filtry libovolně kombinovat. V případě kolize je rozhodující poslední nastavení filtru.

2.1 Odstranění nežádoucích prvků

Privoxy umožňuje odstranit vybrané části webové stránky při zachování hlavního obsahu. Mezi nežádoucí prvky určené k odstranění můžeme zařadit například reklamy. Odstranění některých prvků může mít pozitivní vliv i na bezpečnost, jelikož lze odstranit i potenciálně nebezpečný obsah webové stránky. V této bakalářské práci jsou tyto dva účely odlišeny.

2.1.1 Blokování

Odstranění obsahu pomocí blokování funguje na principu URL adres, respektive IP adres. Privoxy kontroluje adresy v požadavcích klienta, a pokud je požadovaná adresa v seznamu blokovaných adres, je tento požadavek zastaven. Díky tomu nedochází k načtení obsahu blokované webové stránky. Pro výběr blokovaných webových adres je možné použít regulární výrazy. Ty umožňují blokovat jednu konkrétní webovou stránku nebo specifické části domény. Blo-

kovaný obsah je nahrazen zástupným, který je odeslaný přímo z Privoxy. Jako zástupný obsah je standardně použita stránka zdůvodňující zamítnutí požadavku, ale je možné zvolit libovolnou webovou stránku, obrázek nebo prázdný dokument. Podobného výsledku jako při blokování lze dosáhnout i pomocí filtrování obsahu. Jedná se však o dva zcela odlišné postupy.

2.1.2 Omezení obrázků GIF

Obrázky GIF mohou tvořit jednoduchou animaci složenou z několika obrázků zobrazovaných v určitém intervalu. Privoxy umožňuje tyto animace zredukovat na jediný obrázek. Uživatel může GIF obrázek nahradit buďto prvním nebo posledním obrázkem animace. Tím lze eliminovat často rušivé změny obrázků.

2.2 Zvýšení bezpečnosti

Spolu s webovými stránkami může uživatel stáhnout do svého počítače i nebezpečný obsah. Často se o této skutečnosti buď vůbec nedozví, nebo vše zjistí až při prvních problémech s počítačem. Privoxy nabízí několik metod, s jejichž pomocí lze zabránit například spouštění rizikových skriptů či stahování podezřelého obsahu.

2.2.1 Filtrování

Filtrování lze kromě záměny či odstranění textu webových stránek využít i pro odstranění obtěžujících nebo potenciálně nebezpečných prvků v HTML, JavaScriptu nebo v jakémkoliv jiném textovém souboru. Typickým příkladem takových prvků jsou reklamní sdělení nebo skripty spouštěné na straně klienta. Filtry například mohou vyhledat tagy odkazující na aplety jazyka Java nebo na prvky ActiveX a tento odkaz následně odstranit. Aplety se pak nemohou spustit a tím se vylučuje riziko, že by uživatel nevědomky stáhl například trojského koně.[19] Díky tomu má filtrování zejména pro bezpečnost velký význam a z tohoto důvodu je zařazeno právě v této kategorii.

V Privoxy lze filtry aplikovat na veškeré textové soubory a zároveň lze Privoxy přinutit, aby s určitými soubory zacházelo stejně jako s textovými. Pouze soubory s označením text/plain jsou standardně vyjmuty z filtrování, neboť webové servery je často využívají pro označení všech souborů, jejichž typ neznají. Jednotlivé filtry pak lze definovat pomocí regulárních výrazů. Privoxy používá Perl–compatible regular expressions[2], známé též z nástrojů Apache, PHP či z webového prohlížeče Safari[16].

Pro aplikování filtrů je nutné nejprve uložit veškerý obsah filtrovaného souboru do zásobníku. Díky tomu se může uživateli jevit načítání webové stránky zpomalené, jelikož se musí celý obsah načíst a zpracovat filtrem a až

[2]Perl–compatible regular expressions. Dostupné na: `http://www.pcre.org/`

poté jej lze přeposlat webovému prohlížeči. Nedochází tedy k postupnému zobrazování stránky. Privoxy umožňuje načíst soubor o velikosti maximálně 4 MB (lze změnit v konfiguračním souboru). Při překročení této velikosti je soubor odeslán po částech bez uplatnění filtrů. Data šifrovaná pomocí SSL Privoxy filtrovat neumožňuje. Takové rozšíření je součástí implementace této bakalářské práce.

V programu Privoxy již existují předpřipravené filtry, které může uživatel doplnit o vlastní. Předdefinované filtry nabízejí například odstranění reklam podle rozměru, odstranění nevyžádaných vyskakovacích oken nebo předcházení změně rozměrů či pozice okna prohlížeče. K filtrování lze dále využít i aplikaci nebo skript v programovacím jazyku vlastní volby. Tyto externí filtry je pak možné aplikovat na veškerý textový obsah stejně jako filtry definované pomocí prostředků Privoxy. Využití externích filtrů je vhodné v okamžiku, kdy klasické filtry nejsou dostatečně výkonné, neumožňují docílit kýženého efektu nebo jsou pro uživatele příliš obtížné na konfiguraci.

2.2.2 Skrytí typu prohlížeče a operačního systému

Parametr HTTP hlavičky User-Agent informuje webový server o typu webového prohlížeče klienta a jeho operačním systému. Privoxy nabízí filtr pro změnu tohoto parametru na uživatelem předdefinovaný řetězec. U některých webových stránek může touto změnou dojít k problémům, jelikož zaslaný obsah může být přizpůsoben na míru konkrétnímu prohlížeči. Kvalitní webové stránky by ale měly zobrazit svůj obsah korektně bez ohledu na typ prohlížeče nebo operačního systému. Díky typu operačního systému v odesílané hlavičce mohou útočníci využít známých chyb tohoto operačního systému k útoku na klienta. Proto může být vhodné tento parametr upravit.

2.3 Ochrana soukromí

Mnoho webů se snaží získat co nejvíce informací o svých návštěvnících. Tyto informace pak mohou využít například pro cílení reklamy na konkrétního uživatele, k analýze uživatelské přívětivosti a ovladatelnosti navštíveného webu, a k mnoha dalším účelům. Snahou uživatele je často naopak co největší anonymita v prostředí internetu. Privoxy nabízí několik nástrojů, jak ztížit webovým serverům získávání citlivých informací o uživateli, případně jak podstrčit serveru falešné informace.

2.3.1 Odstranění odchozích či příchozích cookies

Cookies jsou malé bloky dat zasílané webovým serverem v hlavičkách protokolu HTTP, které slouží k uložení libovolných informací na straně klienta. Nejčastěji obsahují informace o aktuálním připojení k webovému serveru (například uživatelské jméno či obsah nákupního košíku) nebo o chování uživatele

na webových stránkách. Díky citlivým informacím v jejich obsahu jsou častým cílem útočníků. Zásluhou cookies například není nutné přihlašovat se svým uživatelským jménem a heslem vždy, když chceme zobrazit doručené emaily, nebo zobrazit platby v internetové bance. Cookies udržují informace o našem dřívějším přihlášení, a tudíž ho není nutné opakovat. Pokud však cookies získá útočník, může se dostat například na náš email bez zadání uživatelského jména a hesla.[11][3]

Privoxy umožňuje cookies odstranit z hlaviček HTTP protokolu, a to buď z hlaviček příchozích od webového serveru, nebo z hlaviček požadavků od klienta. Tím sice uživatel ztrácí veškeré výhody vyplývající z cookies, ale na druhé straně je tak ztíženo sledování jeho aktivit na webovém serveru či cílení reklamy. Snižuje se také riziko zneužití cookies případným útočníkem.

2.3.2 Omezení životnosti cookies

Cookies obsahují mimo jiné i informace o délce platnosti (parametr Expires). Privoxy umožňuje omezit dobu platnosti cookies přicházejících od serveru na požadovaný počet minut. Pokud je životnost příchozích cookies již nižší než nastavený limit, ke změně nedochází. Důsledky snížení životnosti cookies pak závisí na webovém serveru. Jestliže server obnovuje cookies s každou odpovědí (dostatečně často vzhledem k životnosti nastavené filtrem), cookies zůstávají platné a případné opětovné přihlášení tak není vyžadováno. Pokud však webový server odešle cookies s velmi dlouhou životností pouze jednou na začátku spojení a nedochází k jejich aktualizaci, na straně klienta vyprší platnost cookies po uplynutí filtrem omezeného času. Poté může být přihlášení znovu požadováno, jelikož by prohlížeč neplatné cookies neměl odesílat webovému serveru.

2.3.3 Pouze session cookies

Session cookies jsou takové cookies, které nemají definovaný parametr Expires. Není tedy uvedena jejich doba platnosti. Z tohoto důvodu jsou považovány za platné až do ukončení relace mezi klientem a serverem, tedy do vypnutí webového prohlížeče, případně uzavření příslušné záložky. Poté by měly být prohlížečem smazány. Některé prohlížeče ale umožňují i obnovu již uzavřených záložek včetně cookies. Ty pak odstraní cookies až spolu s daty potřebnými pro obnovu relace.[11][4] Příslušný filtr tedy odstraňuje parametr Expires pro cookies z hlavičky serveru.

[3] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies`

[4] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie`

2.3.4 Změna parametru If-Modified-Since

Tento filtr umožňuje odstranit parametr If-Modified-Since nebo změnit jeho hodnotu. Změnu lze provést o náhodnou hodnotu ze zvoleného rozsahu. Odstranění tohoto parametru je vhodné, pokud chce uživatel načíst obsah webové stránky znovu, nejen z paměti prohlížeče. To může být užitečné například při testování filtrů obsahu. Náhodnou změnou hodnoty parametru pak lze webovému serveru bránit ve sledování návštěv dané stránky uživatelem.

Parametr If-Modified-Since odesílá klient webovému serveru v hlavičce požadavku spolu s časem. Ten určuje, kdy byla daná webová stránka, kterou má uloženou v paměti prohlížeče, naposledy změněna. Webový server zkontroluje tento časový údaj, a pokud došlo ke změně obsahu požadované webové stránky po tomto okamžiku, odesílá požadovaný obsah. Pokud ke změně nedošlo, server odpoví pouze stavovým kódem 304 Not Modified. Webový prohlížeč poté načte tuto webovou stránku ze své paměti.[11][5]

Problém se sledováním pohybu uživatele vychází z doporučení RFC 2616[7, sekce 14.25], aby jako čas nejnovější verze webové stránky uložené v paměti byl použit čas přijatý od serveru. Tento čas zasílá server v hlavičce odpovědi jako parametr Last-Modified. Důvod pro toto doporučení vyplývá z možnosti rozdílného času klienta a serveru, navíc server může striktně vyžadovat přesnou shodu odeslané a přijaté hodnoty. Díky tomu ale může server klientovi odeslat specifické časové razítko a pokud následně obdrží tuto hodnotu v rámci požadavku klienta, dokáže přesně identifikovat odesilatele a jeho předchozí požadavky. Náhodnou změnou hodnoty se identifikace uživatele pro webový server komplikuje.[12]

Privoxy zároveň nabízí i možnost upravit nebo odstranit parametr Last-Modified v hlavičce odpovědi od serveru. Podobný problém způsobuje také parametr If-None-Match, který pracuje na podobném principu, pouze není přenášeno časové razítko, ale tzv. ETag[11][6]. I pro ten však existuje příslušný filtr na jeho odstranění.

2.3.5 Změna parametru From

Tento filtr chrání uživatele před odesláním emailové adresy klienta na webový server. To hrozí hlavně u starších webových prohlížečů pomocí parametru From v HTTP hlavičce. Pokud filtr nalezne tento parametr, je buďto odstraněn, nebo je emailová adresa nahrazena zvoleným řetězcem. Moderní prohlížeče však tento parametr neodesílají, a tak potřeba tohoto filtru klesá.

[5] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-Modified-Since`

[6] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-None-Match`

2.3.6 Skrytí odkazujícího serveru

Parametr Referer v hlavičce HTTP protokolu může webovému serveru sdělit adresu předchozí webové stránky, ze které na požadovanou stránku přecházíme. Privoxy nabízí filtr pro změnu nebo odstranění tohoto parametru. Pro změnu může být použita například adresa domovské stránky webu, který žádáme o odpověď, případně zvolený text. Pokud je tento parametr zcela odstraněn z hlavičky, domnívá se webový server, že klient napsal požadovanou webovou adresu přímo do prohlížeče. Je vhodné upravovat parametr Referer pouze mezi přechody na jiný web, a ne při změně pouhé stránky v rámci jednoho webu. V opačném případě může vést taková úprava k odlišení konkrétního uživatele od ostatních a následnému trasování jeho aktivity. Tyto informace lze poté jednoduše vyčíst z logů serveru. Nepřetržitým odstraňováním tohoto parametru nebo nastavováním jedné konkrétní hodnoty může dojít k odmítnutí od serveru, ve snaze zabránit vložení nebo linkování z jiného webu.

2.3.7 Přidání vlastní hlavičky

Privoxy umožňuje uživateli definovat vlastní HTTP hlavičku, která bude odesílána vybranému serveru nebo serverům. Je možné přidávat libovolné množství hlaviček. Typicky je tato možnost používána pro zmatení logování či pro specifické potřeby uživatelů. Jedno z konkrétních použití je přidání Do-not-track hlavičky. Tou může uživatel sdělit své preference ohledně sledování jeho aktivity webovým serverem.[21]

2.3.8 Rychlé přesměrování

Některé webové servery, jako například yahoo.com, nenabízejí přímé odkazy na další webové stránky. Místo nich směrují uživatele na skript, kterému je předávána cílová adresa jako parametr. Tento skript pak uživatele mimo jiné přesměruje na cílovou adresu, ale může také například ukládat jednotlivé kroky uživatele. Privoxy nabízí přímé přesměrování na cílovou adresu, pokud v požadované adrese objeví jinou adresu jako parametr. Jak ale samotný manuál zmiňuje, ne vždy musí být odkaz v parametru skutečně cílovou adresou. V takovém případě přesměrování způsobí nedostupnost požadované stránky. Obcházení těchto skriptů může být obranou proti sledování pohybu uživatele webovým serverem, ale také urychlí načtení požadované stránky.

2.3.9 Změna parametru Accept-Language

Parametr Accept-Language sděluje webovému serveru, jaký jazyk pro požadované webové stránky klient preferuje. Tento parametr bývá obvykle nastaven podle jazyka operačního systému, případně podle druhu webového prohlížeče. Z tohoto důvodu může upravení parametru Accept-Language způsobit snadnou odlišitelnost klienta od ostatních, jelikož většina klientů má tento pa-

rametr původní a tudíž shodný. Díky odlišitelnosti pak lze vysledovat pohyb konkrétního uživatele mezi jednotlivými webovými stránkami. Změnou tohoto parametru také uživatel zvyšuje šanci na získání obsahu webové stránky ve zvoleném jazyce, pokud ji webový server v tomto jazyce nabízí. Za tímto účelem je vhodné upravit i další parametry v žádosti. Vhodná změna tohoto parametru nám také může pomoci při maskování identity klienta.

2.4 Zachování kompatibility

Určitými úpravami v přenášených datech lze přizpůsobit komunikaci tak, aby byla zachována kompatibilita pro některé aplikace. Problémy mohou způsobit například novější verze protokolu HTTP. Díky možnosti přiřadit tato opatření pouze určitým webovým serverům není nijak omezována komunikace kompatibilních aplikací.

2.4.1 Snížení verze HTTP protokolu

Protokol HTTP/1.1 může způsobovat problémy některým aplikacím. Tato verze protokolu umožňuje například lepší práci s cache pamětí, a pro tyto účely používá další parametry v HTTP hlavičkách. Starší veze protokolu však těmto parametrům nerozumí. Z důvodu zachování funkčnosti aplikací pracujících pouze se starší verzí HTTP je možné v Privoxy změnit protokol HTTP/1.1 na HTTP/1.0. Toto snížení má však také negativní efekt na rychlost načtení webové stránky, i když by nemělo dojít k úplnému rozbití komunikace. Z tohoto důvodu je doporučeno použít tuto možnost pouze v nutných případech.

2.4.2 Bránění komprimaci obsahu

Webové servery často odesílají obsah zkomprimovaný. Díky tomu jsou kladeny menší nároky na přenosovou kapacitu, ale pro uplatnění filtrů je nutné nejdříve provést dekomprimaci. Odstraněním parametru Accept-Encoding z hlavičky zasílané klientem by mělo být zajištěno, že server nebude odesílaná data komprimovat. Parametrem Accept-Encoding totiž klient sděluje serveru, které algoritmy podporuje. Pokud tento parametr není vyplněn, sděluje klient serveru, že je preferována odpověď bez komprimace. I tak ale server může odeslat odpověď komprimovanou, rozhodnutí závisí zcela na něm.[11][7] Komprimace může velikost některých textových souborů snížit o více než 50 %. Jejím vypnutím tak zpomalíme přenos obsahu. Z tohoto důvodu by mělo být toto opatření používáno pouze v nutných případech.

[7] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept-Encoding`

2.5 Další využití

Některé funkce pro práci s přenášenými daty nelze zcela jednoznačně zařadit do zmíněných kategorií. Tato kapitola se věnuje právě těmto specifickým možnostem. Jejich specificita však nikterak nesnižuje jejich využitelnost v praxi.

2.5.1 Změna parametru Content-Disposition

Parametr Content-Disposition je zasílaný webovým serverem v hlavičce odpovědi. Tento parametr určuje, jakým způsobem má být naloženo s přenášeným obsahem. Obsah může být zobrazen přímo ve webovém prohlížeči (tzv. inline), nebo k němu může být přistupováno jako k příloze, která je určena ke stažení a uložení na disk.[11][8] Privoxy obsahuje filtr, který umožňuje tento atribut odstranit nebo změnit. Díky tomu může být zabráněno stahování nechtěného obsahu na disk počítače.

2.5.2 Změna parametru Content-Type

Parametr Content-Type sděluje prohlížeči typ doručeného souboru. Tento parametr je odesílán v HTTP hlavičce webovým serverem. Podle typu souboru pak prohlížeč rozhoduje, jak bude obsah zpracován. Jednotlivé hodnoty tohoto parametru například určují HTML dokument, CSS dokument nebo GIF obrázek. Tento parametr tak pomáhá prohlížeči se správným interpretováním přijatých souborů. Změna parametru může být vhodná například pokud prohlížeč zobrazuje zdrojový kód skriptu namísto jeho výstupu.

[8] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Disposition`

Kapitola 3

Princip proxy serveru

Tato kapitola se podrobně věnuje principu proxy serverů ve vztahu k protokolům HTTP a HTTPS. Jsou zde popsány zejména jednotlivé odlišnosti při filtrování těchto dvou protokolů v rámci proxy serveru. Pro lepší porozumění této problematice je kapitola doplněna o stručný popis protokolu HTTPS.

3.1 Základní princip

Jak již bylo zmíněno, v knize [19] je princip proxy serveru popsán takto: „Proxy fungují tak, že naslouchají požadavkům o služby od interních klientů a pak je předávají na externí síť jakoby byl klientem – původcem samotný server proxy. Jakmile obdrží proxy od veřejného serveru odpověď, vrátí tuto odpověď původnímu internímu klientskému počítači, jako kdyby byl sám původním veřejným serverem.“ Pokud tedy chce klient získat obsah webové stránky pomocí protokolu HTTP, odešle HTTP požadavek na adresu proxy serveru. Proxy server následně požadavek zpracuje, vyhodnotí jeho oprávněnost podle předem stanovených pravidel, a pokud vyhovuje pravidlům, je požadavek odeslán příslušnému webovému serveru. Po obdržení odpovědi webového serveru proxy zpracuje přijatá data, případně aplikuje své filtry, a následně data odešle klientovi. Tento princip předávání komunikace pomocí prostředníka je nazýván Man in the middle (MITM).[13] Jelikož přenášené informace nejsou jakkoliv chráněny proti čtení neoprávněnými subjekty, nebrání nic v použití filtrů.

3.2 Protokol HTTPS

Počátky protokolu HTTPS sahají do devadesátých let, kdy byly uskutečňovány první elektronické platby přes internet. Mnohé společnosti tehdy začaly hledat řešení na ochranu těchto finančních transakcí. Byla diskutována řada možností, jak chránit přenášená data o platbách před útočníky. Jedním z řešení bylo rozšíření protokolu HTTP o možnost šifrovat a případně elektronicky

podepsat přenášený obsah. Tento návrh byl pojmenován S–HTTP a později oficiálně specifikován v RFC 2660[17]. Navzdory tehdejším předpokladům se toto řešení nestalo dominantním.[13]

3.2.1 SSL/TLS protokol

S odlišným postupem přišla firma Netscape Communications. Ta chtěla umožnit navázání zabezpečeného spojení klient/server namísto klasického. Toho bylo dosaženo přidáním nové vrstvy mezi aplikační a transportní vrstvu modelu TCP/IP. Tato nová vrstva je nazývána vrstva bezpečných soketů (SSL) a stará se nejen o navázání zabezpečeného spojení, ale i o přenos dat přes toto spojení. Samotná vrstva SSL se skládá ze dvou dalších podvrstev a několika protokolů. Jako příklad těchto protokolů lze uvést SSL Handshake Protocol, SSL Change Cipher Spec Protocol nebo SSL Application Data Protocol. Všechny tyto podvrstvy a protokoly jako celek jsou obvykle nazývány SSL protokol. Toto řešení umožňuje bezpečně přenášet libovolný aplikační protokol skrz zabezpečené spojení. Další výhodou je téměř úplná transparentnost pro uživatele, který musí pouze ověřit certifikát serveru. To je však také nejslabší bod tohoto řešení.[13]

Protokol SSL byl postupně inovován v několika verzích a následně z něj byl odvozen protokol TLS. SSL/TLS protokol poskytuje tři základní bezpečnostní služby:

- Důvěrnost spojení

  Přenášené informace nejsou dostupné pro neoprávněné subjekty.

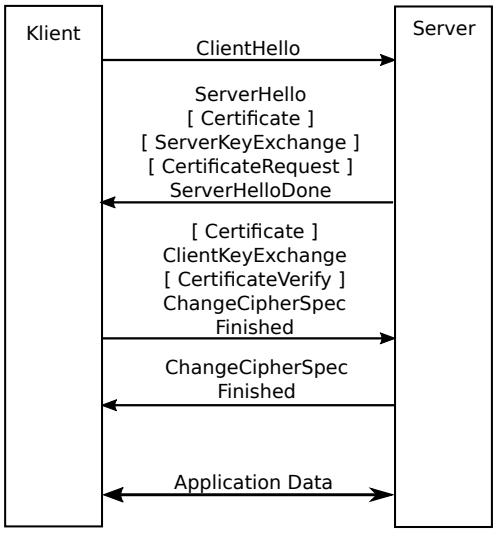
- Autentizace

  Ověření identity subjektu, se kterým je navazováno spojení.

- Integrita spojení (bez obnovy)

  Odhalení neoprávněné změny dat. Pokud k takové změně dojde, je tato změna odhalena. Není však možné tuto změnu obnovit zpět.

Pro zajištění těchto služeb je využívána asymetrická i symetrická kryptografie. Na počátku navazování zabezpečeného spojení probíhá tzv. handshake. Během něj dochází k dohodnutí společných parametrů a výměně dalších potřebných informací. Například je dohodnuta verze SSL/TLS protokolu nebo způsob stanovení společného symetrického klíče. Zároveň server odesílá klientovi svůj certifikát. Dále je stanoven společný klíč pomocí dohodnuté metody. Pro výměnu klíče lze použít například asymetrickou kryptografii pomocí veřejného a soukromého RSA klíče nebo metodu Diffie–Hellman. Po vytvoření společného klíče jsou veškerá přenášená data šifrována tímto symetrickým klíčem. To umožňuje rychlejší šifrování a dešifrování oproti asymetrické kryptografii. Struktura handshake komunikace je zobrazena na obrázku 3.1. Parametry, které nemusí být vždy odesílány, jsou uvedeny v hranatých závorkách.[13]



Obrázek 3.1: Struktura SSL handshake (Překresleno podle [13]).

Protokol HTTPS je tedy protokol HTTP, který je přenášen pomocí zabezpečeného spojení s využitím SSL/TLS protokolu. Díky tomu je obsah HTTP protokolu šifrovaný a tím chráněný proti čtení či úpravám neoprávněnými subjekty. Zároveň dochází k autentizaci webového serveru pomocí certifikátu. Autentizován však může být i klient, pokud to webový server vyžaduje.

3.3 Proxy server a HTTPS

Protokol HTTPS přináší pro proxy servery řadu změn oproti klasickému HTTP protokolu. Tyto změny jsou nutné zejména z důvodu šifrování přenášené komunikace. Šifrováním je proxy serveru znemožněno bez dalších úprav zobrazit obsah komunikace a aplikovat na něj filtry. V této části jsou popsány dva základní postupy používané proxy servery při práci s protokolem HTTPS.

3.3.1 SSL/TLS tunel

Metoda SSL/TLS tunelu umožňuje přenášet zabezpečenou komunikaci skrz HTTP proxy server, aniž by zkoumal obsah přenášených dat či rozuměl jejich struktuře. Klient pouze otevře tzv. tunel v proxy serveru a ten pak předává přijatá data protější straně. Veškerá komunikace mezi klientem a cílovým serverem tedy probíhá skrz proxy server bez jakéhokoliv zásahu. Tato metoda je znázorněna v horní části obrázku 3.2. Z tohoto obrázku je patrné, že při použití SSL/TLS tunelu dochází k vytvoření pouze jediného zabezpečeného spojení, které chrání data na celé trase mezi klientem a cílovým serverem.[13]

Zjednodušený postup při použití metody SSL/TLS tunelu (popsaný v [2] a částečně doplněný o detaily z [13]) je následující:

1. Klient odešle žádost proxy serveru na vytvoření tunelu. Tato žádost musí obsahovat IP adresu (případně doménové jméno) a číslo portu cílového serveru a klienta. Ukázková žádost může vypadat například následovně:

   ```
   CONNECT www.example.com:443 HTTP/1.1
   ```

2. Proxy server přijme žádost a pokusí se navázat TCP spojení s cílovým serverem na požadovaném portu.

3. Pokud se podaří vytvořit TCP spojení s cílovým serverem, potvrdí proxy server klientovi navázání spojení a následně začíná přeposílat veškerou komunikaci mezi klientem a cílovým serverem. V opačném případě informuje klienta o chybě a ukončí s ním spojení.

4. Nyní je řada na klientovi, aby navázal se serverem zabezpečené spojení pomocí SSL/TLS.

5. Veškerá data přijatá proxy serverem od klienta jsou beze změny odeslána na server a data přijatá od serveru jsou ihned odeslána klientovi. Proxy server tak pouze plní funkci prostředníka, který nijak nezasahuje do probíhající komunikace a z důvodu šifrování ani nerozumí struktuře této komunikace. Z tohoto důvodu není možné aplikovat při použití této metody jakékoliv filtrovací mechanismy.

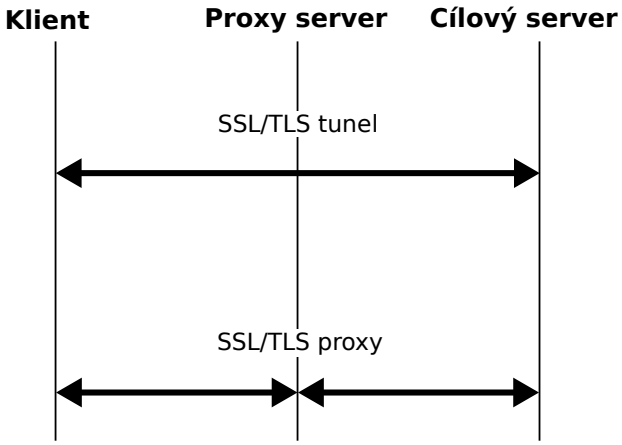
6. Po odeslání všech dat je spojení uzavřeno, nebo může být udržováno pro následující požadavky.

Aby byla tato metoda rozlišitelná i proxy severy, které tunelování nepodporují, musí být požadavek klienta o navázání spojení ve stejném formátu jako klasický požadavek HTTP protokolu. Díky tomu může proxy server upozornit klienta, pokud by takový požadavek nebyl schopný obsloužit.

Dále může proxy server vyžadovat autentizaci klienta. V takovém případě na přijatý požadavek nereaguje okamžitým navazováním spojení s cílovým serverem, ale čeká, dokud klient neprokáže svoji identitu. Podle ní pak může aplikovat případná nastavení. Například povolit pro dotyčného klienta přístup na jinak nepřístupný cílový server.[13]

3.3.2 SSL/TLS proxy

Metoda SSL/TLS proxy umožňuje proxy serveru dešifrovat data přenášená pomocí SSL/TLS protokolu. Tím jsou odstraněny nevýhody SSL/TLS tunelu. Například je možné na přenášená data aplikovat filtry, případně lze kontrolovat, jestli aplikační protokol přenášený skrz zabezpečené spojení je skutečně



Obrázek 3.2: SSL/TLS tunel a proxy (Překresleno podle [13])

HTTPS. Není tedy nutné spoléhat se pouze na číslo portu.[13] Tato metoda však přináší i určité nevýhody. Například nelze použít s klientským certifikátem, aniž by byl dostupný pro proxy server i s privátním klíčem.

Při použití SSL/TLS proxy jsou na rozdíl od tunelu vytvořena dvě šifrovaná spojení. Jedno inicializuje klient a je navázáno s proxy serverem, druhé inicializuje proxy server s cílovým serverem. Tento princip je zobrazen v dolní části obrázku 3.2.[13] Díky tomu může proxy server dešifrovat a opět zašifrovat přenášenou komunikaci, neboť má klíč od obou zabezpečených spojení. Data jsou tak po síti přenášena od klienta až k cílovému serveru šifrovaně, pouze v rámci proxy serveru se data dešifrují a opět zašifrují jiným klíčem.

V některých případech není potřeba šifrovat obě spojení. Například pokud je proxy server umístěný ve stejné síti jako klient, může být tato síť důvěryhodná, a tudíž lze použít HTTP protokol. Případně lze také použít SSL/TLS tunel pro odchozí spojení a SSL/TLS proxy pro příchozí spojení. Tento postup se dnes ale příliš nepoužívá.[13] Teoreticky lze libovolně kombinovat SSL/TLS tunel a proxy, případně šifrovaný a nešifrovaný přenos dat.

Kapitola 4

Implementace

Pro rozšíření programu Privoxy o filtrování HTTPS komunikace je nutné provést řadu dílčích úprav a rozšíření původního zdrojového kódu. Pro některé změny lze současně vybrat z několika různých způsobů realizace. Jednotlivé volby pak mají přímý dopad na funkčnost, bezpečnost či uživatelskou přívětivost výsledného řešení. Tato kapitola popisuje výchozí stav zdrojového kódu a provedené úpravy.

4.1 Původní řešení v Privoxy

Originální verze proxy serveru Privoxy již umožňuje filtrování protokolu HTTP. Pro protokol HTTPS je v původním řešení použita metoda SSL/TLS tunelu. Privoxy tudíž pouze přeposílá šifrovaná data mezi klientem a cílovým serverem a z tohoto důvodu na ně nelze použít již existující filtrovací algoritmy.

Po spuštění Privoxy jsou zpracovány přepínače programu a následně je spuštěn nový proces; původní proces je po vytvoření nového ukončen. Pro potřeby ladění je také možné omezit běh programu na jeden proces s jediným vláknem. Nový proces nejprve načte konfigurační soubor určující parametry proxy serveru, a poté v cyklu čeká na připojení klientů pomocí TCP. Po akceptování TCP spojení s klientem je pro toto spojení vytvořeno nové vlákno (případně proces v závislosti na konfiguraci), které nové spojení obsluhuje. V rámci nového vlákna (respektive procesu) je mimo jiné spouštěna funkce `chat`. Tato funkce se stará například o navázání TCP spojení s cílovým serverem, přeposílání dotazů klienta na webový server a odpovědí webového serveru ke klientovi. Jsou zde také používány filtry pro data přenášená pomocí HTTP. Pokud je používán protokol HTTPS, vytváří tato funkce SSL/TLS tunel mezi klientem a webovým serverem a ten následně obsluhuje. Funkce `chat` je tedy klíčové místo pro rozšíření programu Privoxy o filtrování HTTPS komunikace. Základní struktura upravené funkce `chat` (s vyznačenými změnami oproti původní funkci) je zobrazena v příloze B.

4.2 Man in the middle

Pro filtrování HTTPS komunikace pomocí proxy serveru je nutné přenášená data dešifrovat. Z toto důvodu je potřeba implementovat princip MITM pomocí protokolu SSL/TLS. Poté již proxy server dokáže dešifrovat data přijatá od klienta i od serveru a také je zašifrovat a odeslat dále. Pro práci s SSL/TLS protokolem je nejprve nutné zvolit příslušnou knihovnu. Další volby se týkají například rozsahu nasazení protokolu HTTPS. V následujícím textu je popsáno a zdůvodněno zvolené řešení, včetně dopadů tohoto způsobu řešení na výslednou funkčnost.

4.2.1 Knihovna pro SSL/TLS protokol

Vzhledem ke skutečnosti, že Privoxy používá licenci GNU GPLv2[9], je nutné zvolit příslušnou SSL/TLS knihovnu v souladu s touto licencí. Z tohoto důvodu není možné využít velmi rozšířenou knihovnu OpenSSL[10]. V této bakalářské práci je tedy použita knihovna mbed TLS[11] (původně PolarSSL). Tato knihovna má shodnou licenci jako Privoxy a tudíž ji lze použít při dodržení licenčních podmínek.

4.2.2 Navázání SSL/TLS spojení s cílovým serverem

Původní funkce `chat` nejprve zpracuje přijatý požadavek od klienta, aby určila parametry nutné pro další postup. Poté je navázáno TCP spojení s cílovým serverem. V tomto místě je vhodné doplnit navázání TCP spojení o vytvoření zabezpečeného SSL/TLS spojení, pokud klient žádá o HTTPS komunikaci. Postup však závisí na případném použití nadřazeného proxy serveru.

1. Nadřazený proxy server není použit

   Po navázání TCP spojení s cílovým serverem je na proxy serveru spuštěn SSL/TLS klient, který se pokouší navázat zabezpečené spojení s cílovým serverem. Po navázání tohoto spojení je vytvořena pravá část SSL/TLS proxy z obrázku 3.2.

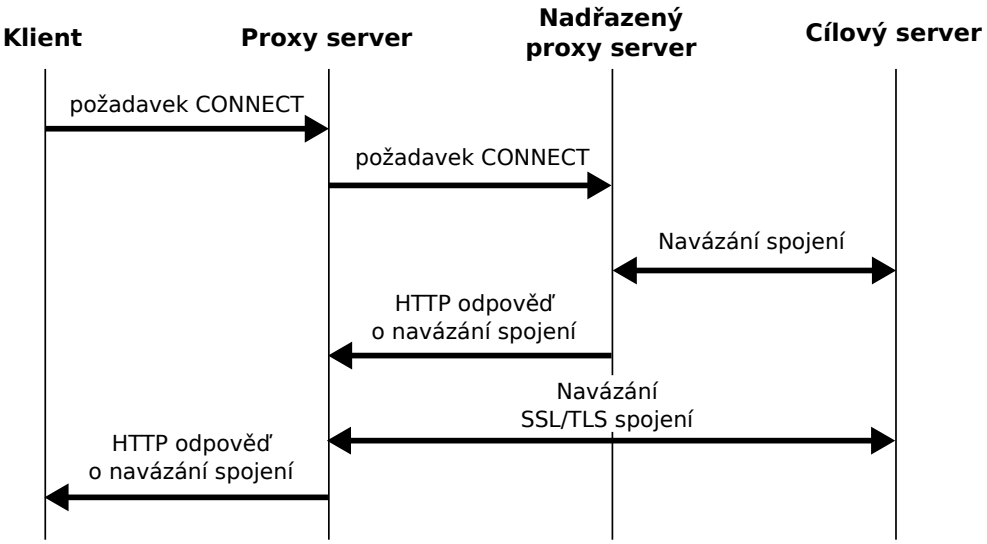
2. Nadřazený proxy server je použit

   V tomto případě proxy server nevytváří spojení přímo s cílovým serverem, ale s nadřazeným proxy serverem, který následně s cílovým serverem komunikuje. Aby nadřazený proxy server získal potřebné informace, je nutné mu ihned po vytvoření TCP spojení přeposlat požadavek CONNECT od klienta. Nadřazený proxy server se poté pokouší připojit

[9] GNU General Public License, version 2. Dostupné na: `https://www.gnu.org/licenses/old-licenses/gpl-2.0.html`

[10] OpenSSL Software Foundation. Dostupné na: `https://www.openssl.org/`

[11] mbed TLS. Dostupné na: `https://tls.mbed.org/`



Obrázek 4.1: Postup navázání zabezpečeného spojení s cílovým serverem při použití nadřazeného proxy serveru

k požadovanému serveru a následně odesílá zpět HTTP odpověď s informacemi o výsledku připojování. Podle RFC 2817[9] odpověď oznamuje úspěšné spojení se serverem, pokud stavový kód začíná číslicí 2. Proxy server se v takovém případě pokusí vytvořit SSL/TLS spojení s nadřazeným proxy serverem (respektive s cílovým serverem, pokud nadřazený proxy server vytváří SSL/TLS tunel). Pokud je i toto zabezpečené spojení úspěšně navázáno, přepošle proxy server klientovi odpověď o úspěšném navázání spojení s cílovým serverem. Pokud se však některé ze spojení nepodaří navázat, je klientovi odeslána informace o chybě a následně jsou veškerá spojení ukončena. Celý postup je znázorněn na obrázku 4.1.

Nabízí se možnost ponechat část spojení mezi proxy a cílovým serverem nezabezpečenou. Tím by byla zachována funkčnost pro některé webové stránky, ale po internetu by byla přenášena nezašifrovaná data od klienta. Současně by některé webové stránky, které neumožňují komunikaci pomocí HTTP, byly pro klienta nedostupné.

4.2.3 Navázání SSL/TLS spojení s klientem

Po vytvoření spojení se serverem je v původním programu postupováno v závislosti na použití SSL/TLS protokolu a nastavení přesměrování. Jednotlivé možnosti jsou zobrazeny v tabulce 4.1. Z této tabulky vyplývá, že ke změnám musí opět dojít v obou částech, kde je použit SSL/TLS protokol.

Tabulka 4.1: Postup Privoxy při navazování spojení

Nadřazený proxy server	SSL/TLS: NE	SSL/TLS: ANO
NE	Je odeslán upravený HTTP požadavek na příslušný webový server. V požadavku je URL adresa nahrazena pouhým umístěním zdroje na serveru. Odpověď od serveru lze filtrovat, a poté odeslat klientovi.	Klientovi je odeslána odpověď „`200 Connection established`“ a následně jsou přeposílána veškerá data mezi klientem a cílovým serverem. Vytváří se tak tzv. SSL/TLS tunel.
ANO	Přijatý HTTP požadavek je odeslán na adresu nadřazeného proxy serveru. V požadavku je zapsána celá URL. Odpovědi od proxy serveru mohou být filtrovány a následně jsou odeslány klientovi.	Přijatý požadavek je beze změny odeslán na určenou adresu. Požadavek CONNECT je tedy doručen nadřazenému proxy serveru. Tento proxy server pak také odesílá odpověď „`200 Connection established`“. Samotné Privoxy takovou odpověď neodesílá, neboť by došlo k jejímu zdvojení. Odpovědi od proxy serveru jsou přeposílány klientovi, nelze je ale filtrovat, jelikož jsou šifrované.

1. Nadřazený proxy server není použit

   Odeslání odpovědi „`200 Connection established`“ klientovi zde zůstává zachováno, jelikož spojení pomocí SSL/TLS protokolu s cílovým serverem je v tuto chvíli již vytvořeno. Poté však musí následovat navázání zabezpečeného spojení s klientem. Proto je na proxy serveru spuštěn SSL/TLS server, který čeká na připojení klienta.

2. Nadřazený proxy server je použit

   V takovém případě je postupováno téměř shodně jako bez použití nadřazeného proxy serveru. Jediný rozdíl spočívá v neodeslání odpovědi „`200 Connection established`“ klientovi, jelikož tato odpověď již byla odeslána během navazování spojení s nadřazeným proxy serverem. Došlo by tak k jejímu zdvojení.

Po navázání zabezpečeného spojení s klientem je již plně dokončen princip MITM pomocí protokolu SSL/TLS. Program tedy může začít přeposílat data mezi klientem a cílovým serverem, přičemž jsou tato data proxy serverem dešifrována. Díky tomu lze aplikovat filtrování i na HTTPS komunikaci.

Teoreticky lze mezi proxy serverem a klientem zachovat klasické TCP spojení a předpokládat, že síť mezi nimi je dostatečně důvěryhodná pro bezpečný přenos dat. V takovém případě by se ovšem webový prohlížeč mohl i nadále pokoušet zabezpečené spojení vytvořit a v případě neúspěchu by spojení s proxy serverem zcela ukončil.

4.2.4 Přenos dat pomocí SSL/TLS

V původním kódu je přenos dat realizován pouze pomocí protokolu TCP. Nyní je ale potřeba rozlišit použití SSL/TLS protokolu a případně komunikovat jeho prostřednictvím. K uložení informací o používání SSL/TLS protokolu je rozšířena stávající struktura `http_request`, která je součástí struktury `client_state`. Strukturu `client_state` získává funkce `chat` jako parametr. Informace v těchto strukturách jsou doplněny mimo jiné podle hlavičky klientova požadavku. Následně je na příslušných místech v kódu testováno použití SSL/TLS protokolu a případně využita nově vytvořená funkce pro odeslání dat tímto protokolem. Zjednodušená ukázka tohoto rozšíření je zobrazena v následujícím kódu:

```
if(client_use_ssl(csp))
{
   ssl_send_data(&(csp->mbedtls_client_attr.ssl),
      (const unsigned char *)buf, len);
   ...
}
else
{
   write_socket(csp->cfd, buf, (size_t)len);
   ...
}
```

4.3 Filtrování komunikace

Implementace filtrování HTTPS komunikace je stěžejní bod této bakalářské práce. Algoritmy sloužící k filtrování textového obsahu jsou v Privoxy již implementovány. Je tedy nutné pouze upravit již existující program tak, aby byly tyto filtry použity i pro HTTPS komunikaci, pokud je jejich použití vyžadováno konfigurací Privoxy.

4.3.1 Zpracování HTTP hlavičky přijaté od serveru

Pro správné použití filtrů je nejprve nutné zjistit informace o přenášené komunikaci. Na jejich základě lze poté rozhodnout, zda má být filtrování uskutečněno a případně jaké filtry použít. Klíčové informace pro toto rozhodnutí se nacházejí v HTTP hlavičce odesílané cílovým serverem na začátku odpovědi. Původní program tuto hlavičku analyzoval pouze při použití protokolu HTTP, protože u protokolu HTTPS nemohl tuto hlavičku dešifrovat. Nyní je ale potřeba upravit zdrojový kód tak, aby byla analyzována HTTP hlavička odpovědi i při použití SSL/TLS protokolu. Je tedy nutné změnit podmínku ve funkci `chat`, která při použití SSL/TLS protokolu ihned odesílala klientovi data přijatá od serveru. Nyní tak není v tomto místě rozlišováno mezi protokolem HTTP a HTTPS. Po načtení HTTP hlavičky od cílového serveru jsou získány veškeré potřebné informace pro další postup a následně je volána původní funkce `content_requires_filtering`. Ta na základě předložených informací rozhodne, zda má být pro následující data použito filtrování.

4.3.2 Aplikování filtrů

Jak již bylo zmíněno, před samotným filtrováním je nutné nejdříve načíst celý obsah přenášeného souboru. K tomu slouží proměnná `iob` (input output buffer) ve struktuře `client_state`. Aby byla veškerá data přenášená pomocí protokolu HTTPS uložena do tohoto zásobníku, je potřeba opět upravit několik podmínek v rámci funkce `chat`. Tyto podmínky určují postup při použití protokolu HTTPS. Nejprve je do zásobníku vložena HTTP hlavička odpovědi, a poté i veškerý následující obsah.[12] Po načtení celého přenášeného souboru do zásobníku (délka dat přijatých funkcí pro čtení je rovna nule) dojde k volání funkce `execute_content_filters`. Tato funkce je volána také v případě, kdy nastane chyba při přenosu dat mezi proxy serverem a cílovým serverem, a v zásobníku `iob` je načtena alespoň část přenášeného souboru. Funkce `execute_content_filters` již je součástí původního programu Privoxy. V rámci této funkce se nejdříve provede dekomprese dat a vyhledání jednotlivých filtrů nastavených pro danou komunikaci. Poté jsou tyto filtry aplikovány na kopii obsahu proměnné `iob` a tato upravená kopie obsahu je návratovou hodnotou funkce. Na upravená data je opět použita komprese a následně je upravena původní HTTP hlavička v závislosti na provedených změnách dat. Takto upravená data včetně hlavičky již lze odeslat klientovi.

4.4 Dynamické vytváření certifikátů

Při použití protokolu HTTPS kontroluje webový prohlížeč identitu webového serveru pomocí certifikátu. Pokud prohlížeč shledá certifikát neplatným, upo-

[12] Velikost zásobníku je určena v konfiguračním souboru a pokud přenášený soubor tuto velikost překročí, k filtrování nedojde a obsah zásobníku je ihned odeslán.

zorní uživatele chybovým hlášením s možností udělit výjimku pro konkrétní certifikát. Kontrolované vlastnosti certifikátu jsou například doba platnosti nebo shoda subjektu uvedeného v certifikátu s doménovým jménem zadaným v prohlížeči. Navíc musí být certifikát podepsán pro prohlížeč důvěryhodnou certifikační autoritou (dále jen CA). Při používání SSL/TLS proxy je navázáno SSL/TLS spojení mezi klientem a proxy serverem, a jelikož je zde proxy v roli serveru, musí odesílat webovému prohlížeči nějaký certifikát k prokázání identity. Pokud by proxy server zasílal klientovi neplatné certifikáty, musel by uživatel pro každou webovou stránku schvalovat výjimku. Tím by klesla použitelnost i bezpečnost celého řešení.

Nabízí se otázka, proč nepřeposlat klientovi originální certifikát cílového webového serveru. To ovšem není možné, jelikož certifikát obsahuje mimo jiné veřejný klíč subjektu, který je využit při navazování SSL/TLS spojení. Proxy server však nemá k veřejnému klíči cílového serveru odpovídající soukromý klíč a tudíž by nemohl dešifrovat data přijatá od klienta, pokud by byla zašifrována tímto veřejným klíčem.

4.4.1 Důvěryhodná certifikační autorita

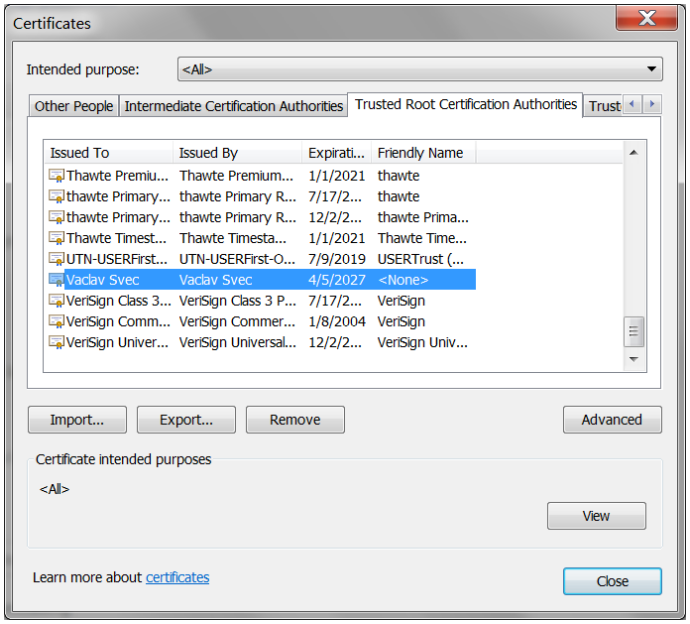
Jako první krok při vytváření certifikátů pro jednotlivé webové stránky je potřeba získat CA. Soukromým klíčem této autority poté lze podepisovat jednotlivé certifikáty pro webové stránky. Zároveň je potřeba tuto CA doinstalovat mezi důvěryhodné CA webového prohlížeče. Díky tomu webový prohlížeč vyhodnotí přijatý certifikát jako důvěryhodný a uživatel již nemusí vytvářet výjimky pro jednotlivé webové stránky. Novou CA lze vytvořit například pomocí následujícího příkazu:

```
#!/bin/bash
openssl req -new -x509 -extensions v3_ca -keyout cakey.pem
   -out cacert.crt -days 3650
```

CA vytvořenou tímto příkazem ale nelze využít pro webový prohlížeč Mozilla Firefox, jelikož certifikáty podepsané takovou CA odmítá uznat jako platné. Přesná příčina tohoto problému mi není známa. Pro prohlížeč Mozilla Firefox je nicméně potřeba vytvořit CA jiným způsobem. Novou CA lze instalovat přímo v nastavení webového prohlížeče. Zde je důležité přidat novou CA mezi důvěryhodné kořenové CA. Zároveň je potřeba nastavit této CA možnost autentizovat webový server. Na obrázku 4.2 je zobrazena nově nainstalovaná CA.

4.4.2 Generování klíče a podpis certifikátu

Při navazování SSL/TLS spojení mezi proxy serverem a klientem je potřeba odeslat klientovi certifikát, který odpovídá doménovému jménu požadovaného cílového serveru. Je tedy nutné takový certifikát buďto vytvořit, nebo použít



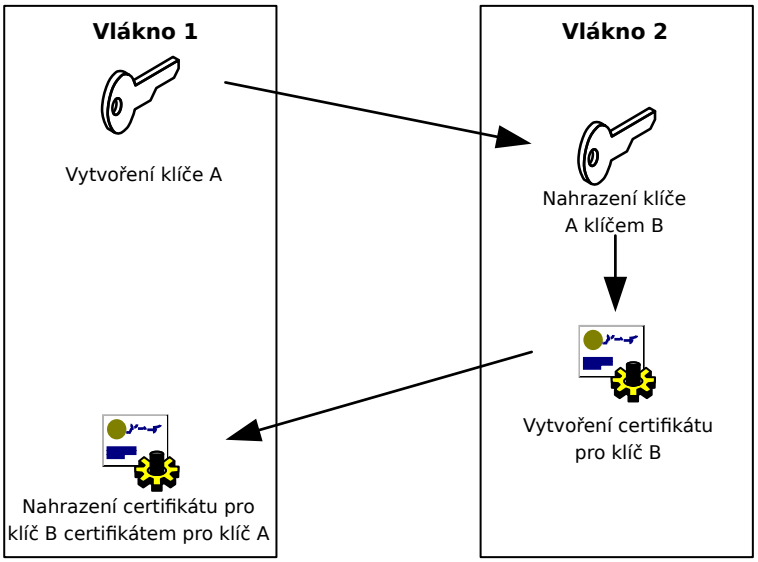
Obrázek 4.2: Vlastní certifikační autorita

již dříve vytvořený. Zároveň je potřeba mít ke každému certifikátu příslušný soukromý klíč. Proto je vytváření zabezpečeného spojení s klientem doplněno o volání funkce, která zajistí vytvoření příslušných klíčů a následných certifikátů ještě před začátkem SSL/TLS spojení. Funkce nejprve kontroluje existenci soukromého klíče ve složce určené pro nově vytvořené certifikáty a jejich klíče. Pokud klíč neexistuje, dojde k jeho okamžitému vytvoření. Následně je nový klíč použit při vytváření certifikátu pro danou webovou stránku. Tento certifikát je podepsán soukromým klíčem vlastní CA.

Pro veškeré certifikáty by bylo teoreticky možné použít pouze jeden pár soukromého a veřejného klíče. Došlo by tak ke zrychlení při načítání webových stránek, jelikož průměrná doba generování jednoho klíče o délce 2048 bitů je přibližně o řád vyšší než doba vytváření certifikátu. Tento způsob ale není v implementaci použit, neboť některé důsledné webové prohlížeče by mohly rozdílné certifikáty se stejným veřejným klíčem označit jako neplatné. Navíc díky ukládání již vytvořených certifikátů a jejich soukromých klíčů dochází k vytvoření certifikátu pro dané doménové jméno pouze při prvním načítání dat z této domény. Poté je již opakovaně používán existující certifikát.

4.4.3 Problémy při generování klíče a certifikátu

Při vytváření nových certifikátů je důležité zabránit chybám, které mohou vzniknout díky běhu ve více vláknech současně. Mohlo by totiž dojít k současnému vytváření klíče a certifikátu pro jedno doménové jméno ve dvou nebo



Obrázek 4.3: Chyba při vytváření certifikátů

více vláknech. V takovém případě by první vlákno vytvořilo soukromý klíč, poté by druhé vlákno tento klíč přepsalo novým a vytvořilo k němu certifikát a následně by první vlákno přepsalo původní certifikát novým, ale podle svého původního klíče. Tato situace je znázorněna na obrázku 4.3. Výsledkem by tedy byla dvojice soukromý klíč a certifikát vydaný pro jiný klíč. Z tohoto důvodu jsou v programu použity mutexy, které zabrání vytváření klíče a certifikátu ve více vláknech současně, ale pouze pokud se jedná o stejná doménová jména. Toho je docíleno díky poli 65 536 mutexů (pomocí makra `LIMIT_MUTEX_NUMBER` lze snížit počet mutexů na 32) a MD5 heše (hash) doménového jména. Ta je vytvořena ještě před generováním klíče a certifikátu a její první dva byty odkazují na konkrétní mutex v poli. Ten je nutné pro pokračování uzamknout. Díky tomu lze vytvářet certifikáty pro různá doménová jména současně, ale je zabráněno výše popsané chybě.

Heš doménového jména je následně používána i pro získání sériových čísel jednotlivých certifikátů. Pomocí makra `CERT_SERIAL_NUM_LENGTH` lze určit počet bytů od začátku heše, které mají být použity pro sériové číslo certifikátu. Pokud by bylo používáno neustále stejné sériové číslo pro všechny certifikáty podepsané jednou CA, webový prohlížeč by mohl označit tyto certifikáty za neplatné. Alternativně by bylo možné použít soubor pro ukládání posledního použitého sériového čísla. Toto řešení by ovšem vyžadovalo další použití mutexů pro čtení a zápis tohoto souboru. Heš doménového jména je rovněž používána jako název pro ukládání vytvořených klíčů a certifikátů. Tím je předcházeno problémům, pokud by doménové jméno obsahovalo znaky, které se nemohou vyskytovat v názvech souborů použitého souborového systému.

4.5 Kontrola certifikátů na straně proxy serveru

Při použití SSL/TLS proxy ověřuje webový prohlížeč certifikát proxy serveru namísto certifikátu cílového serveru (případně nadřazeného proxy serveru). Z tohoto důvodu musí být certifikát cílového serveru kontrolován proxy serverem, který vytváří SSL/TLS spojení přímo s cílovým serverem, a tudíž získává jeho certifikát. Pokud by proxy server neověřoval platnost certifikátu cílového serveru, mohl by klientovi odeslat odpověď od nedůvěryhodného serveru. Webový prohlížeč by pak certifikát proxy serveru vyhodnotil jako platný a uživatel by o neplatném certifikátu cílového serveru nebyl nijak varován.

4.5.1 Ověření certifikátu

Pro ověření certifikátu webového serveru je nezbytné získat seznam důvěryhodných CA. V této bakalářské práci je použit seznam CA exportovaný z webového prohlížeče Mozilla.[13] Pokud není certifikát cílového serveru podepsaný jednou z důvěryhodných CA ze seznamu, je tento certifikát označen jako neplatný. Další kontrolované vlastnosti webového certifikátu jsou například doba platnosti nebo název subjektu, pro který byl certifikát vytvořen. V rámci práce ovšem není z časových důvodů implementována kontrola revokovaných certifikátu ani podpora pro tzv. pinning veřejných klíčů[14].

4.5.2 Odhalení neplatného certifikátu

Kontrola certifikátu cílového serveru probíhá během SSL handshake. Pokud je certifikát shledán neplatným, je handshake ukončen. Následně je potřeba podrobně informovat klienta o důvodu přerušení komunikace s cílovým serverem. Proto je klientovi nejdříve odeslána odpověď:

```
HTTP/1.1 200 Connection established
```

Ta klientovi potvrzuje úspěšné navázání spojení s cílovým serverem. Následně je s klientem navázáno SSL/TLS spojení. Bez těchto kroků by proxy server nemohl odeslat klientovi další podrobnosti o přerušení spojení. Po navázání SSL/TLS spojení proxy server odesílá klientovi následující odpověď:

```
HTTP/1.1 200 OK
Content-Type: text/html
Connection: close

<html><body>
    Podrobnosti o certifikatu ciloveho serveru
</body></html>
```

[13] Dostupné na: `https://curl.haxx.se/ca/cacert.pem`

[14] Více informací o této problematice na: `https://developer.mozilla.org/en-US/docs/Web/HTTP/Public_Key_Pinning`

Obrázek 4.4: Informace o neplatném certifikátu

Součástí odesílaných informací o certifikátu cílového serveru jsou mimo jiné i jednotlivé certifikáty z jeho řetězce, které je možné po načtení webové stránky prohlížečem stáhnout. Certifikáty jsou do webové stránky vloženy pomocí Data URI schématu, které umožňuje vkládat tyto certifikáty přímo mezi HTML kód. Pouze je nutné certifikát zakódovat pomocí Base64 kódování. Vložený kód pro stažení certifikátu pak vypadá takto:

```
<a href="data:application/x-x509-ca-cert;base64,
   LS0tLS1CRU...0tLS0tDQo=">
Download certificate</a>
```

Na obrázku 4.4 je zobrazena vizuální podoba výsledné webové stránky, která je odeslána klientovi. Z obrázku je také patrné, že certifikát použitý pro SSL/TLS spojení mezi klientem a proxy serverem je z pohledu webového prohlížeče platný. To dokazuje zelený zámek v poli určeném pro URL adresu. Po odeslání odpovědi je SSL/TLS spojení s klientem ukončeno a poté je ukončena i funkce `chat`. V ten okamžik je vlákno obsluhující původní spojení uvolněno pro nové požadavky.

Alternativním řešením, jak informovat klienta o neplatném certifikátu webového serveru, je použití neplatného certifikátu. Při navazování SSL/TLS spojení mezi proxy serverem a klientem by proxy server použil předem připravený certifikát určený k tomuto účelu. Certifikát by byl například podepsaný pro webový prohlížeč neplatnou CA. Tím by byl aktivován mechanismus pro kontrolu certifikátů na straně webového serveru, který by klienta na neplatný certifikát upozornil a spojení pozastavil. Při použití tohoto postupu by ovšem

proxy server neměl jakoukoli možnost předat klientovi podrobné informace o certifikátu cílového serveru. Navíc pokud by uživatel pro certifikát určený k informování o neplatnosti certifikátu udělil výjimku, bylo by spojení mezi klientem a proxy serverem navázáno. Otázkou ovšem je, jaká data by měl v takové chvíli proxy server klientovi odesílat, neboť uživatel udělil výjimku, aniž by znal podrobné informace o certifikátu cílového serveru. Navíc by tato výjimka platila pro veškerá následující spojení, kde proxy server odhalí neplatný certifikát. Tato metoda tedy obnáší řadu bezpečnostních nedostatků, a proto v bakalářské práci není implementována.

4.5.3 Testování spolehlivosti

Pro ověření správné detekce neplatných certifikátů proxy serverem byly použity ukázkové certifikáty z webové stránky `https://badssl.com/`. V tabulce 4.2 je zapsáno, které chyby a nedostatky (nejen certifikátů) jsou proxy serverem odhaleny. Zároveň je zde srovnání výsledků s webovými prohlížeči. Z tabulky je patrné, že základní nedostatky certifikátů umí proxy server odhalit a bezpečně na ně klienta upozornit. Pokročilejší metody pro kontrolu certifikátů ovšem nejsou v proxy serveru implementovány a tudíž nejsou problémy tohoto typu odhaleny.

4.6 Možnosti konfigurace

Vzhledem k rozšíření funkcionalit programu Privoxy je vhodné tyto funkcionality také alespoň částečně konfigurovat bez nutnosti úpravy zdrojových kódů a následné kompilace. Proto program umožňuje definovat nové parametry z konfiguračních souborů.

4.6.1 Obecná konfigurace SSL/TLS

Obecné vlastnosti proxy serveru lze konfigurovat v souboru `config`. V rámci implementace je tento soubor rozšířen o nová klíčová slova, díky kterým lze konfigurovat základní vlastnosti SSL/TLS spojení. Přehled těchto slov a jejich účel je zobrazen v tabulce 4.3. K načítání nových parametrů je použit stejný způsob jako pro veškeré další, které již byly v původním programu Privoxy implementovány. K načtení změn tedy dochází po spuštění programu a následně periodicky před každým otevřením nového připojení.

4.6.2 Konfigurace akcí

V souboru `user.action` lze aktivovat funkcionality programu Privoxy a případně je přiřadit ke konkrétní množině webových stránek. I v tomto souboru je možné v rámci rozšíření využít dvě nové funkcionality. Klíčové slovo `disable-ssl-filtering` umožňuje pro vybrané webové stránky stahované

Tabulka 4.2: Srovnání výsledků Privoxy a webových prohlížečů při odhalování chyb a nedostatků SSL/TLS spojení. (Ano – odhaleno, Ne – neodhaleno)

Test	Privoxy	Google Chrome[15]	Mozilla Firefox[16]	IE[17]
Certifikát – expired	Ano	Ano	Ano	Ano
Certifikát – wrong.host	Ano	Ano	Ano	Ano
Certifikát – self-signed	Ano	Ano	Ano	Ano
Certifikát – unrusted-root	Ano	Ano	Ano	Ano
Certifikát – revoked	Ne	Ano	Ano	Ano
Certifikát – pinning-test	Ne	Ano	Ano	Ne
Certifikát – no-common-name	Ne	Ne	Ne	Ne
Certifikát – no-subject	Ne	Ne	Ne	Ne
Certifikát – incomplete-chain	Ano	Ne	Ne	Ne
Certifikát – sha1-intermediate	Ne	Ne	Ano	Ne
Certifikát – invalid-expected-sct	Ne	Ano	Ne	Ne
Smíšený obsah – mixed-script	Ne	Ne	Ne	Ne
Smíšený obsah – very	Ne	Ne	Ne	Ne
Smíšený obsah – mixed	Ne	Ne	Ne	Ne
Smíšený obsah – mixed-favicon	Ne	Ne	Ne	Ne
Smíšený obsah – mixed-form	Ne	Ne	Ne	Ne
Šifrovací sada – cbc	Ne	Ne	Ne	Ne
Šifrovací sada – rc4-md5	Ano	Ano	Ano	Ano
Šifrovací sada – rc4	Ano	Ano	Ano	Ano
Šifrovací sada – 3des	Ne	Ne	Ne	Ne
Šifrovací sada – null	Ano	Ano	Ano	Ano
Výměna klíče – dh480	Ano	Ano	Ano	Ano
Výměna klíče – dh512	Ano	Ano	Ano	Ano
Výměna klíče – dh1024	Ne	Ano	Ne	Ne
Výměna klíče – dh2024	Ne	Ano	Ne	Ne
Výměna klíče – dh-small-subgroup	Ne	Ano	Ne	Ne
Výměna klíče – dh-composite	Ne	Ano	Ne	Ne
Výměna klíče – static-rsa	Ne	Ne	Ne	Ne
Aktualizace – subdomain.preloaded-hsts	Ano	Ano	Ano	Ano
Známé špatné – Superfish	Ano	Ano	Ano	Ano
Známé špatné – eDellRoot	Ano	Ano	Ano	Ano
Známé špatné – DSD Test Provider	Ano	Ano	Ano	Ano
Zaniklé – sha1-2016	Ano	Ano	Ano	Ano
Zaniklé – sha1-2017	Ano	Ano	Ano	Ano

pomocí HTTPS využít původní metodu SSL/TLS tunelu. Tím zároveň dochází k vypnutí všech filtrů pro takto přenášená data. Druhé klíčové slovo `ignore-certificate-errors` umožňuje vypnout kontrolu certifikátů cílových serverů pro vybrané webové stránky. Díky tomu lze ručně definovat určitý seznam výjimek pro případné neplatné certifikáty. V rámci bakalářské práce ovšem není umožněno vytvořit výjimku pouze pro určitý parametr certifikátu (například dobu platnosti) a ostatní i nadále kontrolovat.

[15] Google Chrome verze 57.0.2987.133 (64 bitů)
[16] Mozilla Firefox verze 53.0 (32 bitů)
[17] Internet Explorer verze 11.0.9600.18638 (64 bitů)

Tabulka 4.3: Nová klíčová slova v konfiguračním souboru `config`

Klíčové slovo	Účel
`ca-password`	Nastavení hesla soukromého klíče CA
`ca-dir`	Nastavení adresáře obsahujícího CA, její klíč a důvěryhodné CA
`ca-cert-file`	Nastavení názvu souboru CA
`ca-key-file`	Nastavení názvu souboru s klíčem CA
`trusted-cas-file`	Nastavení názvu souboru s důvěryhodnými CA
`certs-dir`	Nastavení adresáře pro ukládání vytvořených certifikátů a jejich klíčů pro jednotlivé webové stránky

4.7 Zhodnocení dosažených výsledků

Zadání této bakalářské práce požaduje implementaci základní verze filtrování HTTPS komunikace. Do jaké míry byl splněn tento požadavek a jaké konkrétní funkcionality jsou nebo nejsou v rámci práce implementovány, je shrnuto v této části. Zároveň jsou zde naznačena i další vhodná vylepšení.

4.7.1 Funkčnost

Výsledný program úspěšně provádí metodu MITM. Tím je umožněno dešifrovat veškerá data přenášená mezi klientem a webovým serverem pomocí protokolu HTTPS. Následně lze použít vlastní definované filtry pro úpravu veškerých souborů přicházejících od webového serveru. Kromě vlastních filtrů lze použít i další předdefinované filtry, které například blokují příchozí cookies, umožňují libovolně filtrovat hlavičku serveru, upravují vybrané parametry hlavičky serveru nebo omezují animace souborů GIF. Uživateli je tak umožněno například odstranit z webových stránek reklamní sdělení, nahradit existující části stahovaných souborů jinými, nebo odstranit odkazy na potenciálně nebezpečné soubory, čímž je zabráněno jejich stažení a spuštění. Všechny tyto funkce lze využít ke zvýšení bezpečnosti, a to nejen při použití protokolu HTTP, ale i u protokolu HTTPS. Funkce pro úpravu hlavičky serveru či odstranění příchozích cookies lze dále využít pro ochranu soukromí uživatele.

Současná verze programu však neumožňuje využít naprosto všechny filtrovací možnosti Privoxy. Konkrétně nelze uplatnit většinu filtrů na požadavky přijaté od klienta. Tento nedostatek vyplývá ze struktury funkce `chat`, která nepředpokládá filtrování HTTPS protokolu a její struktura tudíž nedovoluje takové filtrování jednoduše implementovat. Funkce `chat` umožňuje filtrovat pouze první přijatý požadavek ze strany klienta a další data jsou již pouze přeposílána bez jakýchkoli úprav. Při použití protokolu HTTPS je však první přijatý požadavek typu CONNECT, který nežádá o konkrétní webovou stránku, ale o vytvoření spojení s cílovým serverem. Požadavek CONNECT je tedy možné částečně filtrovat, ale následující požadavek obsahující klíčové informace již nikoli. Pro odstranění tohoto nedostatku by bylo vhodné buďto vytvořit celou funkci `chat` znovu tak, aby počítala s možností filtrovat HTTPS

protokol, nebo upravit funkci `chat` spolu s dalšími funkcemi pro zpracování hlavičky přijaté od klienta. Vzhledem k tomu, že funkce `chat` obsahuje přes 1000 řádek a ve zdrojovém kódu jsou požadavky na její přepracování, jeví se první varianta jako vhodnější způsob řešení. Z důvodu velkého rozsahu již tyto úpravy nejsou součástí implementace této bakalářské práce.

Funkčnost upraveného programu Privoxy je ověřena na webových prohlížečích Google Chrome[18] a Mozilla Firefox[19] a Internet Explorer[20]. S těmito prohlížeči program spolupracuje bez zjevných problémů. Mírné rozdíly jsou pouze v rychlosti načítání webových stránek nebo četnosti výskytu chyb při navazování spojení. To je způsobeno patrně různou implementací a nastavením jednotlivých prohlížečů. Zároveň je ověřena funkčnost proxy serveru na operačních systémech Windows i Linux. Pro bezproblémové používání je ale nutné doinstalovat CA Privoxy do konkrétního prohlížeče.

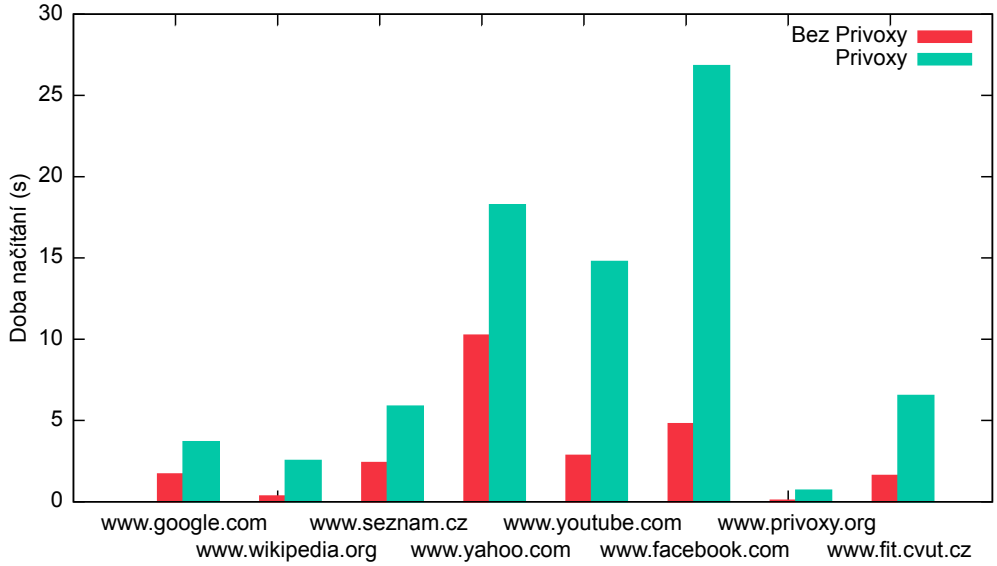
Během implementace této bakalářské práce vyšla nová verze 58.0.3029.96 webového prohlížeče Google Chrome. Tato verze u certifikátů webových stránek striktně vyžaduje nastavení parametru Subject Alternative Name. Implementace bakalářské práce tento parametr nenastavuje, jelikož používaná knihovna mbed TLS ve verzi 2.4.0 nastavení tohoto parametru při generování certifikátu neumožňuje. Z tohoto důvodu jsou všechny certifikáty programu Privoxy označeny webovým prohlížečem Google Chrome verze 58.0.3029.96 jako neplatné. V budoucnu by knihovna mbed TLS již možnost nastavení tohoto parametru měla obsahovat. Následně tedy bude vhodné implementovat[21] nastavení parametru Subject Alternative Name pro vytvářené certifikáty.

Z hlediska stability program nevykazuje žádné problémy. K pádům programu nedochází. Výjimku tvoří řádně logované kritické chyby, pro které nelze pokračovat v běhu programu. V takových případech je běh programu korektně ukončen. Tyto chyby se týkají například načítání konfigurace proxy serveru. O chybách vzniklých při práci se zabezpečeným spojením je klient ve většině případů informován pomocí HTTP odpovědi s příslušným stavovým kódem. Nejsou zde ale použity předdefinované webové stránky programu Privoxy, jelikož původní implementace nepočítala s použitím těchto stránek pro protokol HTTPS. Pro jejich použití by tak byly nutné rozsáhlé změny původních funkcí.

4.7.2 Výkonnost

Načítání webových stránek při použití upraveného programu Privoxy je podle očekávání pomalejší než bez něj. Ve většině případů se však nejedná o tak veliké zpoždění, kvůli kterému by byl program nepoužitelný v praxi. Rozdíly v době načítání vybraných webových stránek pomocí protokolu HTTPS při použití Privoxy a bez něj jsou zobrazeny v grafu na obrázku 4.5. Měření je

[18] Google Chrome verze 57.0.2987.133 (64 bitů)
[19] Mozilla Firefox verze 53.0 (32 bitů)
[20] Internet Explorer verze 11.0.9600.18638 (64 bitů)
[21] Změny se týkají funkce `generate_webpage_certificate` v souboru `ssl.c`.



Obrázek 4.5: Doba načítání vybraných webových stránek

jen orientační, jelikož výkon nebyl požadován a lze předpokládat, že možnost filtrování ho převáží. Graf zobrazuje dobu od odeslání požadavku do načtení většiny obsahu dané webové stránky.[22] Všechna měření probíhala za stejných podmínek (stejný počítač, webový prohlížeč, internetové připojení, stav paměti prohlížeče, předpřipravené certifikáty webových stránek, ...). Z grafu je patrné prohloubení rozdílů při načítání rozsáhlejších webových stránek. To je z velké části způsobeno neefektivním zacházením s SSL/TLS spojeními. Upravený program Privoxy totiž neumí udržovat již vytvořená SSL/TLS spojení pro další dotazy, a tudíž je nutné tato spojení vytvářet pro každý dotaz znovu. Vytváření spojení trvá značnou část z celkové doby komunikace a při načítání velkého množství menších souborů se tak značně zvýší celková doba načítání.

Další příčinou zpomalení mohou být občasné chyby vznikající při vytváření SSL/TLS spojení. Nejčastěji se z pohledu webového prohlížeče jedná o „`ERR_TUNNEL_CONNECTION_FAILED`“ a „`ERR_CONNECTION_RESET`“. V některých případech mohou tyto chyby zabránit načtení celé webové stránky nebo její části. Důvodem těchto chyb může být mimo jiné omezení maximálního počtu otevřených spojení. Při použití HTTPS dochází k jejich vyčerpání a následnému čekání na uvolnění některého z nich. Při použití paměti prohlížeče pro uložení načtených webových stránek jsou však dopady chyb zanedbatelné.

[22] Z pohledu uživatele je doba načítání webové stránky nižší, jelikož většina viditelných částí je načtena dříve, než uplyne doba znázorněná v grafu 4.5.

4.7.3 Bezpečnost

Důležitým aspektem implementovaného rozšíření je jeho bezpečnost. Pro otestování bezpečnostních rizik spojených s protokolem SSL/TLS byla použita řada z testů dostupných na stránkách `www.badssl.com`. Výsledky všech použitých testů jsou zobrazeny v tabulce 4.2. Ze srovnání s webovými prohlížeči vyplývá, že webové prohlížeče odhalí větší množství bezpečnostních rizik, i když některá z nich nejsou kritická (jedná se především o rizika, která jsou odhalena pouze některými ze srovnávaných webových prohlížečů). Hlavní nedostatky proxy serveru jsou v kontrole revokovaných certifikátů. Tu na rozdíl od proxy serveru provádí všechny srovnávané webové prohlížeče. Tento nedostatek již byl zmíněn dříve. V rámci implementace je použito výchozí nastavení pro šifrovací sady i pro protokoly sloužící k výměně klíče. Tato nastavení neumožňuje program upravovat pro potřeby uživatelů. Některá neodhalená rizika by bylo možné ošetřit právě pomocí ručního nastavení.

4.8 Možná budoucí vylepšení

V rámci této bakalářské práce byla implementována řada změn a rozšíření původního programu Privoxy. Vzhledem k rozsahu celého programu je však stále řada možností, jak proxy server vylepšit. V této části jsou popsány některé z nich.

4.8.1 Rychlejší načítání webových stránek

Pro rychlejší načítání webových stránek je vhodné rozšířit program tak, aby efektivněji pracoval s SSL/TLS spojeními. Pokud by byla po určitou dobu udržována již navázaná SSL/TLS spojení pro případné další dotazy na stejný cílový server, nemuselo by tak často docházet k jejich vytváření. Tím by byla uspořena velká část z celkové doby načítání webové stránky. Částečně by pro tuto funkci mohl být využit princip pro udržování TCP spojení, který již původní verze Privoxy obsahuje.

4.8.2 Nové funkcionality

Z pohledu funkcionalit programu je pravděpodobně nejdůležitějším budoucím rozšířením možnost filtrovat nejen všechna data přijatá pomocí protokolu HTTPS od serveru, ale i veškeré dotazy přijaté tímto protokolem od klienta. Jak již bylo zmíněno dříve, implementace v rámci této bakalářské práce umožňuje filtrování dotazů přijatý pomocí HTTPS jen velmi omezeně. S možností využít veškeré filtrovací možnosti programu Privoxy na všechny přijaté dotazy se pojí rozsáhlé změny funkce `chat`, případně rozsáhlé změny již existujících funkcí pro zpracování dotazu klienta. Po implementování těchto změn by však bylo možné využít veškeré filtry programu Privoxy i pro HTTPS komunikaci.

4.8.3 Nové možnosti nastavení

Filtrování protokolu HTTPS přináší celou řadu nových parametrů a možností. Bylo by vhodné umožnit uživatelům proxy serveru tyto vlastnosti konfigurovat podle vlastních potřeb. Jako příklad lze uvést definici filtrů pro dané webové stránky pouze při použití konkrétního protokolu. Uživatel by tak mohl některé filtry spojit pouze s protokolem HTTP, nebo HTTPS. Dalším vhodným rozšířením je také možnost definovat parametry certifikátu pro jednotlivé webové servery. Uživatel by tak mohl pro různé webové servery definovat například různou dobu platnosti certifikátu, případně odlišnou CA pro podpis certifikátu.

4.8.4 Zabezpečení

Vytváření SSL/TLS spojení s cílovými servery přináší řadu možností pro rozšíření s ohledem na bezpečnost. Například je vhodné program rozšířit o kontrolu revokovaných certifikátů, případně o možnost definovat šifrovací sadu programu a schválené algoritmy pro výměnu klíče. Uživatel by tak nemusel spoléhat na výchozí nastavení knihovny mbed TLS, ale mohl by si konkrétní parametry přizpůsobit vlastním potřebám.

Zároveň je vhodné v případě odhalení neplatného certifikátu umožnit uživateli udělení výjimky přímo z prostředí webového prohlížeče. Současná implementace umožňuje tuto výjimku vytvořit pouze pomocí úpravy souboru `user.action`, což je nepraktické a zdlouhavé řešení. Navíc by bylo vhodné umožnit tuto výjimku specifikovat pouze pro konkrétní nedostatek (například pro dobu platnosti certifikátu) a další vlastnosti i nadále kontrolovat.

Závěr

Cílem této bakalářské práce bylo analyzovat způsoby filtrování HTTPS komunikace, seznámit se s problematikou proxy serverů ve vztahu k HTTP/HTTPS a na základě získaných informací poté navrhnout a implementovat rozšíření programu Privoxy. Toto rozšíření by mělo umožnit základní filtrování HTTPS komunikace pomocí filtrovacích nástrojů programu.

V kapitolách jedna a dvě jsou popsány způsoby a možnosti filtrování HTTPS komunikace. Kapitola tři je věnována problematice proxy serverů ve vztahu k protokolům HTTP/HTTPS. Na základě těchto poznatků se podařilo úspěšně navrhnout a implementovat rozšíření využívající princip Man in the middle (MITM), díky kterému lze dešifrovat HTTPS komunikaci. Následně jsou použity filtrovací nástroje původního programu Privoxy pro filtrování dat přijatých pomocí HTTPS od cílového serveru. Toto filtrování lze navíc využít i při použití nadřazeného proxy serveru. Filtrování dotazů přijatých pomocí HTTPS od klienta je však velmi omezené, jelikož původní program Privoxy není pro takové filtrování přizpůsobený a nutné úpravy původních zdrojových kódů by byly rozsáhlé. Kromě principu MITM a zapojení filtrů jsou v rámci rozšíření implementovány i další podpůrné části nezbytné pro bezpečné a pohodlné používání proxy serveru.

Přestože již nyní umožňuje Privoxy filtrovat HTTPS komunikaci, je zde stále řada příležitostí pro další rozšíření. Program by bylo vhodné rozšířit například o udržování navázaných SSL/TLS spojení, možnost udělit výjimku pro neplatné certifikáty nebo o možnost použít veškeré filtry na dotazy přijaté od klienta.

Díky této práci jsem se podrobněji seznámil s řadou témat vyučovaných napříč mnoha předměty. Zejména se jedná o témata z oboru webových technologií, počítačových sítí a bezpečnosti. Úpravami programu Privoxy se mi navíc podařilo značně rozšířit jeho využitelnost a výsledný program může být výchozím bodem pro jeho další vývoj.

Literatura

[1] Anderson, N.: *Deep packet inspection meets 'Net neutrality, CALEA*. [online], červenec 2007, [cit. 19.2.2017]. Dostupné z: `https://arstechnica.com/gadgets/2007/07/deep-packet-inspection-meets-net-neutrality/`

[2] Ariyarathna, P.: *What is SSL Tunneling?* [online], listopad 2015, [cit. 5.12.2016]. Dostupné z: `https://dzone.com/articles/what-is-ssl-tunneling`

[3] Bradley, M.: *What is a DNS Server?* [online], listopad 2016, [cit. 13.2.2017]. Dostupné z: `https://www.lifewire.com/what-is-a-dns-server-817513`

[4] Cheswick, W. R.; Bellovin, S. M.; Rubin, A. D.: *Firewalls and Internet Security: Repelling the Wily Hacker*. Boston: Addison-Wesley, druhé vydání, 2003, ISBN 0-201-63466-X.

[5] Crocker, S.; Shinkuro; Verisign; aj.: *Security and Other Technical Concerns Raised by the DNS Filtering Requirements in the PROTECT IP Bill*. [online], květen 2011, [cit. 14.2.2017]. Dostupné z: `http://domainincite.com/docs/PROTECT-IP-Technical-Whitepaper-Final.pdf`

[6] Deibert, R.: *Access denied: the practice and policy of global Internet filtering*. MIT Press, 2008, ISBN 978-0-262-54196-1.

[7] Fielding, R.; aj.: *Hypertext Transfer Protocol – HTTP/1.1*. [online], červen 1999, [cit. 19.3.2017]. Dostupné z: `https://www.w3.org/Protocols/rfc2616/rfc2616.html`

[8] Kbelová, A.; Dostálek, L.: *Velký průvodce protokoly TCP/IP a systémem DNS*. Brno: Computer Press, páté vydání, 2008, ISBN 978-80-251-2236-5.

[9] Khare, R.; Lawrence, S.: *Upgrading to TLS Within HTTP/1.1.* [online], květen 2000, [cit. 20.4.2017]. Dostupné z: `https://www.ietf.org/rfc/rfc2817.txt`

[10] Lawrence, E.: *Selectively Filtering Content in Web Browsers.* [online], listopad 2010, [cit. 27.2.2017]. Dostupné z: `https://blogs.msdn.microsoft.com/ie/2010/11/30/selectively-filtering-content-in-web-browsers/`

[11] Mozilla Developer Network and individual contributors: *HTTP.* [online], únor 2017, [cit. 15.3.2017]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/HTTP/`

[12] Musing Mortoray: *How HTTP cache headers betray your privacy.* [online], květen 2015, [cit. 14.3.2017]. Dostupné z: `https://mortoray.com/2015/05/11/how-http-cache-headers-betray-your-privacy/`

[13] Oppliger, R.: *SSL and TLS: theory and practice.* Artech House, 2009, ISBN 978-1-59693-447-4.

[14] Parsons, J. J.: *Computer concepts. New perspectives series.* Cengage Learning, 2016, ISBN 978-1-305-38775-1.

[15] Privoxy: *Privoxy 3.0.26 User Manual, Actions Files.* [online], [cit. 13.3.2017]. Dostupné z: `https://www.privoxy.org/user-manual/actions-file.html`

[16] Privoxy: *Privoxy 3.0.26 User Manual, Appendix.* [online], [cit. 30.3.2017]. Dostupné z: `https://www.privoxy.org/user-manual/appendix.html`

[17] Rescorla, E.; Schiffman, A.; aj.: *The Secure HyperText Transfer Protocol.* [online], srpen 1999, [cit. 30.3.2017]. Dostupné z: `https://tools.ietf.org/html/rfc2660`

[18] Sosinsky, B.: *Mistrovství – počítačové sítě.* Brno: Computer Press a.s., 2009, ISBN 978-80-251-3363-7.

[19] Strebe, M.; Perkins, C.: *Firewally a proxy-servery: Praktický průvodce.* Brno: Computer Press a.s., 2003, ISBN 80-7226-983-6.

[20] Walker, H. M.: *The TAO of COMPUTING.* Hoboken: CRC Press, druhé vydání, 2013, ISBN 978-1-4398-9252-7.

[21] World Wide Web Consortium: *Tracking Preference Expression (DNT).* [online], 2015, [cit. 14.3.2017]. Dostupné z: `https://www.w3.org/TR/tracking-dnt/`

[22] Zwicky, E. D.; Cooper, S.; Chapman, D. B.: *Building Internet Firewalls: Internet and Web security. 2. edice*. Cambridge, Mass.: O'Reilly, druhé vydání, červen 2000, ISBN 1-56592-871-7.

Příloha A

Seznam použitých zkratek

CA Certifikační autorita

DNS Domain Name System

HTTP Hyper–Text Transfer Protocol

HTTPS Hyper–Text Transfer Protocol Secure

IE Internet Explorer

ISP Internet Service Provider

MITM Man In The Middle

RSA asymetrická šifra autorů Rivest, Shamir, Adleman

SSL Secure Sockets Layer

TCP Transmission Control Protocol

TLS Transport Layer Security

URI Uniform Resource Identifier

URL Uniform Resource Locator

Příloha B

Struktura funkce `chat`

Následující pseudokód zobrazuje velmi zjednodušenou strukturu funkce `chat` včetně provedených úprav. Části upravené nebo přidané během implementace této bakalářské práce jsou označeny pomocí komentářů. Z důvodu rozsahu celé funkce zde nejsou zobrazeny veškeré úpravy.

```
void chat(struct client_state *csp){
  receive_client_request(client_header);
  parse_client_request(client_header);

  open_tcp_connection_with_server();

  /* Pridano: Navazani SSL/TLS spojeni s~cilovym serverem nebo
              nadrazenym proxy serverem */
  if(ssl && !use_ssl_tunel){
    if(parent_proxy_server){
      write_socket(server_socket, client_header);
      read_socket(server_socket, server_response);

      if(!tunel_established_successfully(server_response)){
        write_socket(client_socket, server_response);
        return;
      }

      create_server_ssl_connection();
      write_socket(client_socket, server_response);
    }else{
      create_server_ssl_connection();
    }
  }
```

```
/* Upraveno: Puvodne if(parent_proxy_server || !ssl) */
if(!ssl || (parent_proxy_server && use_ssl_tunnel)){
  write_socket (server_socket, client_header);
}else{
  /* Upraveno: Puvodne pouze
    write_socket(client_socket,
       "200 connection established");  */
  if(use_ssl_tunnel){
    write_socket(client_socket,
       "200 connection established");
  }else{
    if(!parent_proxy_server){
      write_socket(client_socket,
         "200 connection established");
    }

    create_client_ssl_connection();
    if (server_cert_invalid){
      ssl_send_certificate_error(client_connection);
      return;
    }
  }
}

server_body = 0;
for(;;){
  /* Pridano: Kontrola SSL/TLS spojeni s klientem, jestli
              od nej necekaji data ke cteni. */
  if(client_want_communicate || ssl_pending(client_ssl)){
    if( !client_use_ssl){
      read_socket(client_socket);
      write_socket(server_socket);
    }else{
      ssl_recv_data(client_ssl);
      ssl_send_data(server_ssl);
    }
  }

  /* Pridano: Kontrola SSL/TLS spojeni se serverem, jestli
              od nej necekaji data ke cteni. */
  if(server_want_communicate || ssl_pending(server_ssl)){
    /* Upraveno: Jine prijimani dat pri pouziti SSL/TLS
                 spojeni a bez nej. */
    if( !server_use_ssl){
```

```
  len = read_socket(server_socket, buffer);
}else{
  len = ssl_recv_data(server_ssl, buffer);
}

if(len < 0){
  return;
}

if(len == 0){
  /* Upraveno: Puvodne if(server_body || ssl) */
  if(server_body || (ssl && use_ssl_tunnel)){
    if(buffer_and_filter_content){
      filter_buffer = execute_content_filters(iob);
      pdate_server_headers(csp);

      /* Upraveno: Jine odesilani dat pri pouziti SSL/TLS
                   spojeni a bez nej. */
      if( !client_use_ssl(csp)){
        write_socket(client_socket, header);
        write_socket(client_socket, filter_buffer);
      }else{
        ssl_send_data(server_ssl, header);
        ssl_send_data(server_ssl, filter_buffer);
      }
    }
  }
}

/* Upraveno: Puvodne if(server_body || ssl) */
if(server_body || (ssl && use_ssl_tunnel)){
  if(buffer_and_filter_content){
    add_to_iob(iob, buffer);
  }
}else{
  /* Upraveno: Jine odesilani dat pri pouziti SSL/TLS
               spojeni a bez nej. */
  if( !client_use_ssl(csp)){
    write_socket(client_ssl, buffer);
  }else{
    ssl_send_data(client_ssl, buffer);
  }
  continue;
}else{
```

```
        get_server_headers(header);

        filter_server_headers(header);
        buffer_and_filter_content =
           content_requires_filtering(csp);

        if (!buffer_and_filter_content)
        {
          /* Upraveno: Jine odesilani dat pri pouziti SSL/TLS
                       spojeni a bez nej. */
          if( !client_use_ssl(csp)){
            write_socket(client_ssl, header);
          }else{
            ssl_send_data(client_ssl, header);
          }
        }
        server_body = 1;
      }
      continue;
    }
    return;
  }
  return;
}
```

Příloha C

Uživatelská příručka

C.1 Požadavky pro instalaci

Pro přeložení zdrojového kódu programu Privoxy včetně implementovaného rozšíření jsou potřeba nástroje autoconf, GNU make (gmake) a kompilátor jazyka C (například gcc). Po rozbalení souboru zip je potřeba v kořenovém adresáři projektu zajistit existenci adresáře mbedtls obsahujícího zkompilovanou knihovnu mbed TLS. Zdrojový kód poté lze přeložit pro operační systém Linux nebo Windows (k tomuto účelu lze využít například nástroj Cygwin). Postup pro dosažení požadovaného stavu může být následující:

```
tar -xvzf ./implementation.tar.gz
cd ./implementation
tar -xvzf ./mbedtls-2.4.0-gpl.tgz
mv ./mbedtls-2.4.0 ./mbedtls
cd ./mbedtls
make
cd ..
```

C.2 Instalace

Kompilace a instalace programu může být provedena pomocí následující sekvence příkazů:

```
autoheader && autoconf && ./configure && make
make -s install
```

Před spuštěním programu Privoxy je ještě nutné vytvořit potřebnou adresářovou strukturu a CA pro proxy server. Toho lze docílit například pomocí příkazů níže. Ty předpokládají výchozí nastavení parametrů pro SSL v konfiguračním souboru `config`. V tomto souboru je navíc potřeba nastavit heslo vytvořeného soukromého klíče, pokud je tento chráněn heslem. CA vytvořená

tímto způsobem ale není dostačující pro webový prohlížeč Mozilla Firefox. Pro tento prohlížeč je potřeba vytvořit CA jiným postupem.

```
mkdir CA
mkdir certs
cd CA
openssl req -new -x509 -extensions v3_ca -keyout cakey.pem
   -out cacert.crt -days 3650
cd ..
```

C.3 Nastavení webového prohlížeče

Po dokončení všech předchozích kroků již lze spustit program Privoxy. Pro správnou spolupráci s webovým prohlížečem je ale ještě nutné upravit nastavení prohlížeče. Aby webový prohlížeč s proxy serverem spolupracoval správně, je ještě potřeba doinstalovat CA a nastavit IP adresu a port proxy serveru. Pokud tyto parametry nejsou v konfiguračním souboru `config` upraveny, naslouchá proxy server na adrese `127.0.0.1:8118`.

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... Stručný popis obsahu CD
- `exe`
  - `build-Windows.tar.gz` ..... Spustitelná forma implementace
- `src`
  - `changed_src_files.tar.gz` ..... Pouze upravené zdrojové kódy
  - `implementation.tar.gz` ..... Zdrojové kódy implementace
  - `thesis.tar.gz` ..... Zdrojová forma práce ve formátu LaTeX
  - `original_source_files`
    - `mbedtls-2.4.0-gpl.tgz` ..... Knihovna mbed TLS
    - `privoxy-3.0.26-stable-src.tar.gz` ..... Výchozí zdrojové kódy
- `text`
  - `BP_Vaclav_Svec_2017.pdf` ..... Text práce ve formátu PDF