VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

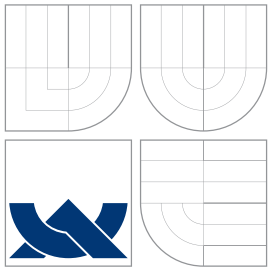
IMPLEMENTACE AUTENTIZACE COSIGN V PHP

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

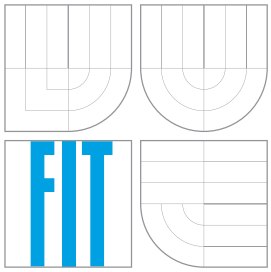
AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ KOVÁŘÍK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IMPLEMENTACE AUTENTIZACE COSIGN V PHP

COSIGN AUTHENTICATION IN PHP

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ KOVÁŘÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR LAMPA

BRNO 2008

Abstrakt

Diplomová práce je zaměřena na problematiku centrální autentizace webových serverů pomocí technologie cookie. Popsány jsou současné metody poskytující jednotné přihlášení. Podrobně je prozkoumána specifikace single sign-on mechanismu Cosign a jeho autentizačního filtru. Popsány jsou kryptografické algoritmy filtru a možnost jejich realizace v jazyce PHP. Práce se dále zabývá samotnou implementací autentizačního filtru, jeho testováním a možným využitím.

Klíčová slova

autentizace, centrální autentizační služba, jednotné přihlášení, cookie, kryptografie, kerberos, cosign, PHP, výkonnost, testování

Abstract

Master's thesis deals with issue of cookie-based central authentication services. Present-day methods of single sign-on are described. The specification of single sign-on mechanism Cosign and its authentication filter is closely viewed. Cryptographic algorithms needed by this filter are described, as well as their possible realization in PHP. Next, the implementation of Cosign authentication filter is described. Performance of the filter is tested and its future use is analysed.

Keywords

authentication, central authentication service, single sign-on, cookie, cryptography, kerberos, cosign, PHP, performance, testing

Citace

Jiří Kovářík: Implementace autentizace Cosign v PHP, diplomová práce, Brno, FIT VUT v Brně, 2008

Implementace autentizace Cosign v PHP

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana ing. Petra Lampy

.....
Jiří Kovářík
19. května 2008

Poděkování

Na tomto místě bych chtěl poděkovat Ing. Petru Lampovi za ochotu k vedení této práce a za cenné připomínky k vznikajícímu textu. Dále bych chtěl poděkovat mé rodině a blízkým za podporu a povzbuzení při psaní této diplomové práce.

© Jiří Kovářík, 2008.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Zadání

1. Prostudujte metody centrální autentizace Web serverů založené na cookie.
2. Na základě dokumentace protokolu autentizace Cosign zhodnoťte potřebné kryptografické funkce pro implementaci klienta protokolu a jejich realizovatelnost v prostředí jazyka PHP.
3. Implementujte knihovnu zajišťující autentizaci protokolem Cosign v PHP.
4. Zhodnoťte výkonnostní parametry implementace v PHP v porovnání s implementací na straně Web serveru v jazyce C.

Část požadovaná pro obhajobu SP: První dva body zadání.

Kategorie: Web

Implementační jazyk: PHP

Operační systém: Linux

Návrhová metodologie: OOP

Volně šířený software: Cosign, Apache, PHP

Literatura: http://www.umich.edu/~umweb/software/cosign/

Komentář: Podmínkou přihlášení tohoto projektu je výstup v podobě kvalitního open source modulu, které bude předatelný umich.edu. Zdrojový kód musí být čitelný, komentovaný v angličtině, instalační dokument v angličtině.

Obsah

Kapitola 1

Úvod

1.1 Motivace

Fakulta informačních technologií (FIT) Vysokého učení technického v Brně uvedla v akademickém roce 2007/2008 do provozu systém webové centrální autentizace. Jedná se o implementaci mechanismu známého jako single sign-on (jednotné přihlášení). Single sign-on (SSO) je metoda kontroly přístupu, která umožňuje klientu pomocí jednoho autentizačního procesu přístup ke všem službám, pro které má klient přístupová práva, bez nutnosti opětovné autentizace. FIT nasadila do provozu implementaci Cosign, která je vyvíjena na Michiganské Univerzitě. Podrobná specifikace produktu je popsána na webových stránkach univerzity[3]. Tato implementace SSO autentizuje uživatele pomocí centrálního webového serveru a následně poskytuje webovému prostředí proměnnou s uživatelským kódem. Když uživatel přistupuje ke službě, která vyžaduje autentizaci, přítomnost této proměnné mu poskytuje přístup bez nutnosti opětovného zadávání přihlašovacích údajů.

Pokud je třeba přístup na jistou sekci webové prezentace zabezpečit filtrem Cosign, je nutné doplnit aktuální verzi serveru Apache o modul `mod_cosign`. Tento modul je napsán v jazyce C, je proto nutné jej nejdříve zkompilovat, nainstalovat a nakonfigurovat server Apache tak, aby při startu modul aktivoval. Ne každý administrátor webových prezentací má ovšem přístup či práva ke změnám na Apache serveru, bylo by tedy přínosné, přepsat filtr také do jazyka PHP, který se používá ke generování dynamických webových stránek. Takovýto PHP modul by jednoduchým vložením do požadované stránky a následnou úpravou konfiguračního souboru nahradil stávající řešení. Před samotnou implementací tohoto modulu je však nutné prozkoumat specifikaci modulu filtru a zhodnotit prostředky a techniky potřebné pro jeho realizaci. V případě úspěšného vybudování modulu filtru bude též nutné provést testování rychlosti zpracování požadavků tímto novým modulem.

1.2 Zpracovaná problematika

Druhá kapitola vysvětluje čtenáři základní pojmy, využívané v pozdějších částech diplomové práce. Detailně je popsán pojem autentizace a její nejčastější metody. Vysvětleno je také fungování a použití cookies. Je zde nastíněn proces jednotného přihlášení (single sign-on) a jeho výhody a nevýhody. Důležitou částí je vysvětlení protokolu Kerberos, na jehož principu funguje cookie centrální autentizace a jehož mechanismu některé autentizační metody přímo využívají.

Ve třetí kapitole jsou popsány různé implementace SSO přístupů. Zvlášť podrobně je

přistoupeno k mechanismu Cosign, jehož autentizační filtr má být implementován v jazyce PHP pro efektivnější využití správci dynamických webových systémů.

Čtvrtá kapitola vysvětluje důležitost použití moderní kryptografie a její dvě hlavní větve - symetrickou a asymetrickou. Vytváří tím podmínky pro pochopení problematiky mechanismu SSL/TLS, který je na moderní kryptografii postaven. SSL/TLS mechanismus je využit systémem Cosign a je v této kapitole taktéž podrobně popsán. Kvůli implementaci autentizačního filtru jsou uvedeny kryptografické funkce, které nabízí jazyk PHP a které byly při realizaci použity.

Pátá kapitola popisuje přípravu k implementaci Cosign filtru a implementaci samotnou. Příprava Cosign infrastruktury je nutná pro samotný běh a komunikaci filtru s ostatními Cosign prvky. Je třeba po jeho vývoj a testování. Dále je v této kapitole popsána vyvinutá knihovna a třídy které obsahuje. U každé třídy je uveden její účel a výčet metod a atributů, které nabízí. Zmíněna je též uživatelská a programátorská dokumentace.

Šestá kapitola se věnuje testování funkčnosti a výkonnosti filtru. Otestovány byly všechny nabízené konfigurační direktivy a popsán jejich vliv na výsledné chování filtru. Též byla testována funkčnost filtru v Cosign infrastruktuře jako celku, podpora škálovatelnosti systému a jednotného přihlašování. Výkonnostní testy byly provedeny v porovnání s testy modulu napsaného v jazyce C pro server Apache.

Závěrečná kapitola shrnuje dosažené poznatky a hodnotí jejich přínos práce pro fakultu, vývojový tým systému Cosign a jeho potenciální budoucí uživatele.

Diplomová práce navazuje na semestrální projekt, jehož úkolem byla teoretická příprava a pochopení dané problematiky. Semestrální projekt pokrývá obsah kapitol 2 až 4.

Kapitola 2

Základní pojmy

2.1 Autentizace

Pojem autentizace je pro potřeby softwarového inženýrství přesně definován podle doporučení RFC[8]. Autentizace je postup, při kterém se něco (nebo někdo) ustanovuje nebo potvrzuje jako autentické. Autentizace objektu může znamenat potvrzení jeho původu, kdežto k autentizaci osoby se často používá ověření její identity. V prostředí informačních systémů je autentizace chápána především jako proces ověření digitální identity odesilatele dat pomocí požadavku na přihlášení (login). Odesilatelem, který je autentizován, může být osoba užívající počítač, počítač samotný nebo počítačový program.

Velká část internetových webových služeb nabízí zřízení uživatelského účtu uchovávajícího osobní či tajné informace. Pro přístup k těmto informacím je nutné, aby se uživatel prokázal kombinací přihlašovacího jména a hesla. Autentizace za použití hesla je nejběžnější, jiné formy prokázání totožnosti zahrnují skenování otisku prstů, sítnice oka či tvaru dlaně, v neposlední řadě se využívá i vlastnictví unikátních objektů (hardwarové tokeny, platební karty). Výhodou „něčeho, co daný uživatel zná“, tedy hesel, je, že se nejedná o fyzický objekt, ale o abstraktní znalost, kterou lze snadno přenášet, zadávat do počítače. Systém pro tuto metodu autentizace lze snadno ovládat a nevyžaduje složitou údržbu. Nevýhodou pak je, že tajná informace může být snadno zjištěna, a to dokonce bez vědomí uživatele. Navíc lidská paměť je s ohledem na zapamatování "náhodných" informací poměrně omezená (složitá hesla si lze jen velmi obtížně zapamatovat), což negativně ovlivňuje celkovou bezpečnost této autentizační metody.

Poté, co se uživatel úspěšně autentizuje, je třeba, aby pro opětovný přístup k zabezpečeným zdrojům nemusel znovu zadávat své přihlašovací údaje. Řešením je vytvoření tzv. autentizačního tokenu, který je uživateli přidělen. Autentizační token (typicky textová informace) je při každém pokusu o přístup k zabezpečenému zdroji automaticky verifikován a pokud nesouhlasí, uživatel je považován za neautentizovaného a musí se znovu přihlásit k systému.

2.1.1 Umístění autentizačního tokenu

Protože je protokol HTTP, který se používá k výměně hypertextových dokumentů, bezestavový, je nutné pro uchování autentizačního tokenu použít jiný, vhodnější mechanismus. Nabízí se předávání autentizačního tokenu v každém HTTP požadavku jako parametr, toto řešení je však málo používané z důvodu přímé viditelnosti takto předávaného tokenu, a z toho plynoucí možnosti zneužití (zkopírování URL adresy včetně tokenu). Takto předávaný

token taktéž zaniká ve chvíli, kdy uživatel opouští webové stránky, či ukončuje činnost prohlížeče. Jiným vhodnějším prostředkem je použití HTTP cookies. HTTP cookies (web cookies, cookies) jsou textové řetězce zasílané serverem webovému prohlížeči a poté zasílané v nezměněné formě webovým prohlížečem při každém přístupu na server.

Účelem autentizačního tokenu je oznámení skutečnosti, že uživatel s tímto tokenem je přihlášen. Nejjednodušším řešením by bylo pojmenovat token `logged_in` a nastavit jeho hodnotu na `true`. Jelikož jsou ale cookies na straně klienta přístupné ke čtení i zápisu, bylo by snadné vytvořit si takovéto cookie i bez znalosti uživatelského jména a hesla, a vydávat se tak za přihlášeného uživatele. Často používanou metodou, jak toto obejít, je přiřadit do cookie při úspěšném přihlášení autentizační token ve formě náhodného řetězce vygenerovaného serverem. Tento náhodně vygenerovaný řetězec je uložen i na serveru a při každém požadavku klienta je hodnota zaslaného autentizačního tokenu v cookie porovnána s hodnotou na serveru. Jestliže se uživatel odhlásí, je autentizační token na serveru smazán.

Stále však hrozí reálné riziko, že mohou být při požadavku na chráněný zdroj odeslaná data odposlechnuta a bude provedena autentizace pomocí odposlechnutého tokenu bez nutné znalosti přihlašovacího jména a hesla. Ani za pomoci přídavných technik jako kontrola IP adresy (lze snadno obejít) či „solení hesel“ nelze garantovat naprosto bezpečný přenos citlivých dat po síti.

Řešením tohoto problému je komunikace přes zabezpečný kanál (Secure Socket Layer, dále jen SSL). SSL protokol dovoluje aplikacím komunikovat skrz síť způsobem, který znemožňuje odposlouchávání, ovlivňování či podvržení přenášených dat. SSL poskytuje za použití kryptografie autentizaci koncových bodů a zachování soukromí při komunikaci přes Internet. SSL je podrobněji popsáno v sekci 4.4, protože k jeho pochopení je nutné blíže osvětlit problematiku moderní kryptografie.

2.2 Cookies

Protože je Cosign technologií využívající cookies, je namístě vysvětlit jejich účel a princip činnosti. Přesná specifikace cookies je uvedena v doporučení RFC[10].

Cookies jsou technikou, jak uchovávat stav při použití HTTP v kontextu HTTP požadavků a odpovědí. HTTP protokol v současné době pracuje tak, že HTTP server odpovídá každému klientu bez návaznosti na předcházející či následující požadavky. Mechanismus řízení stavu dovoluje klientům a serverům, které si potřebují vyměnit stavové informace, umístit HTTP požadavky a odpovědi do širšího kontextu zvaného „session“ (sezení). Ani po klientech, ani po serverech není vyžadováno, aby podporovali cookies. Server nemusí poskytnout obsah klientovi, který nevrátil cookie serverem zaslané.

2.2.1 Syntaxe

Následující dvě hlavičky pro správu stavu, `Set-Cookie2` a `Cookie`, mají společné syntaktické vlastnosti zahrnující páry tvaru atribut-hodnota. Syntaxi popisuje následující gramatika zapsaná v Backus-Naur formě, jak ji doporučuje požívat dokument RFC[11]

```
av-pairs    =    av-pair *(";" av-pair)
av-pair     =    attr ["=" value]              ; volitelná hodnota
attr        =    token
value       =    token | quoted-string
```

Atributy (jména) jsou case insensitive (nezáleží na velikosti písmen). Neviditelné znaky mezi tokeny jsou povoleny. Přestože syntaxe popisuje hodnoty atributů jako volitelné, většina atributů je vyžaduje.

2.2.2 Role zdrojového serveru

- Pokud je třeba, zdrojový server iniciuje sezení. Aby tak učinil, vrací navíc v odpovědi hlavičku klientovi, `Set-Cookie2`.
- Pokud si uživatelský agent přeje pokračování sezení, vrací hlavičku Cookie požadavku zdrojovému serveru. Zdrojový server ji může ignorovat, či použít k zjištění aktuálního stavu sezení.
- Server může poslat klientovi zpět `Set-Cookie2` odpověď, nebo už nemusí posílat žádnou `Set-Cookie2` hlavičku.
- Zdrojový server efektivně ukončí sezení zasláním hlavičky `Set-Cookie2` s hodnotou `Max-Age=0`.
- Servery mohou vrátit `Set-Cookie2` hlavičku v jakékoliv odpovědi.
- Uživatelský agent by měl zaslat cookie request hlavičku při každém požadavku.
- Zdrojový server smí vkládat více `Set-Cookie2` hlaviček v jedné odpovědi.

Syntaxe pro `Set-Cookie2` odpověď je popsána následovně:

```
set-cookie      =       "Set-Cookie2:" cookies
cookies         =       1#cookie
cookie          =       NAME "=" VALUE *(";" set-cookie-av)
NAME            =       attr
VALUE           =       value
set-cookie-av   =       "Comment" "=" value
                |       "CommentURL" "=" <"> http_URL <">
                |       "Discard"
                |       "Domain" "=" value
                |       "Max-Age" "=" value
                |       "Path" "=" value
                |       "Port" [ "=" <"> portlist <"> ]
                |       "Secure"
                |       "Version" "=" 1*DIGIT
portlist        =       1#portnum
portnum         =       1*DIGIT
```

`NAME=VALUE` *Požadované.* Jméno stavové informace (”cookie”) je NAME, a její hodnota je VALUE.

`Comment=value` *Volitelné.* Protože mohou být cookies určeny k ukládání či získání důvěrných informací, hodnota Comment atributu umožňuje zdrojovému serveru popsat jak se má nakládat s cookie hodnotou.

`CommentURL="http_URL"` *Volitelné.* Stejné jako předchozí volba, na adrese http_URL je uložen textový řetězec.

`Discard` *Volitelné.* Atribut udává uživatelskému agentu povinnost smazat cookie okamžitě po skončení činnosti.

`Domain=value` *Volitelné.* Hodnota Domain atributu specifikuje doménu, pro kterou je cookie platné.

`Max-Age=value` *Volitelné.* Hodnota Max-Age určuje životnost cookie v sekundách (desítkové nezáporné celé číslo).

`Path=value` *Volitelné.* Hodnota atributu Path specifikuje podmnožinu URL adres na zdrojovém serverru, na které se cookie aplikuje.

`Port["portlist"]` *Volitelné.* Atribut Port obsahuje seznam portů, na které smí být cookie request požadavek vrácen.

`Secure` *Volitelné.* Atribut Secure dává uživatelskému agentu instrukce, aby při zpětném zasílání tohoto cookie na server použil bezpečnou metodu, z důvodu ochrany autentizace a důvěrnosti informace uložené v cookie.

`Version=value` *Povinné.* Hodnota atributu version, desetinné číslo, určuje verzi specifikace pro stavové řízení, ke které cookie náleží.

2.3 Single sign-on

Single sign-on (SSO), neboli jednotné přihlášení, je způsob kontroly přístupu, který umožňuje uživateli, aby se pouze jednou autentizoval a zároveň získal přístup k více softwarovým systémům. V homogenní IT infrastruktuře nebo tam, kde existuje centralizovaná databáze uživatelů, je SSO výrazným přínosem. Všichni uživatelé v takovéto infrastruktuře mohou mít jeden přihlašovací údaj, se kterým se přes jeden přihlašovací mechanismus dostanou ke všem povoleným zdrojům.

Webové SSO pracuje striktně s aplikacemi, ke kterým se přistupuje přes webový prohlížeč. Požadavek k přístupu na webový zdroj je zachycen buď součástí webového serveru, nebo aplikací samotnou. Neautentizovaní uživatelé jsou přesměrování na autentizační službu a automaticky službou vráceni k požadovanému zdroji pouze po úspěšném přihlášení.

Central Authentication Service (CAS) je SSO protokol navržený tak, aby umožňoval neprověřeným webovým aplikacím autentizovat uživatele pomocí důvěryhodné třetí strany, v našem případě pomocí centrálního serveru. CAS protokol zahrnuje klientský webový prohlížeč, aplikaci požadující autentizaci a CAS server, vůči kterému je autentizace vyžadována. Když klient navštíví chráněnou aplikaci, je automaticky přesměrován aplikací na CAS server. CAS server ověří uživatelské jméno a heslo uživatele prostřednictvím zapezpečené databáze, typicky adresářové služby. Jestliže je uživatelské jméno a heslo platné, přesměruje CAS server klienta zpět na chráněnou aplikaci s přiděleným tokenem. Aplikace otevře TLS spojení přímo s CAS serverem a poskytne mu její vlastní identifikátor služby a token. Jestliže je token pro danou službu platný (uživatel má právo k ní přistupovat), zasílá CAS server aplikaci uživatelské jméno.

2.4 Kerberos

Většina SSO pracuje podobně jako protokol Kerberos[5]. Protokol Kerberos byl navržen na Massachusetts Institute of Technology[1] pro poskytnutí bezpečné autentizace skrz otevřenou a nezabezpečenou síť, ve které mohou být zprávy zasílané mezi účastníky zachyceny.

2.4.1 Terminologie

Key Distribution Center (KDS) - Uchovává tajné klíče (kryptografické klíče) pro "principals", neboli účastníky. Vytváří a distribuuje session klíče (kryptografické klíče). KDC využívá symetrickou kryptografii. Skládá se ze dvou hlavních částí: Ticket Granting Service (TGS) a Authentication Service (AS).

Authentication Service (AS) - Zajišťuje autentizaci.

TGS (Ticket Granting Service) - Vytváří a distribuuje objektům (principals) lístky obsahující session klíč.

Principal (účastník) - Objekt, například uživatel, aplikace, služba nebo zdroj používající Kerberos autentizaci. KDC má na starosti jednu nebo více oblastí (realms) takovýchto objektů. Účastník musí důvěřovat KDC, ale nemusí přímo důvěřovat jeden druhému. Pouze KDC smí uchovávat kopie tajných klíčů každého účastníka.

Service Server (SS) - Služba, ke které může účastník přistupovat

Realm (oblast) - Logicky vybraná skupina účastníků.

Ticket (lístek) - Digitální autentizační token zaslaný z AS. První ticket zaslaný z AS účastníkovi se nazývá Ticket Granting Ticket (TGT).

Secret klíče a Session klíče - Klíče symetrické kryptografie určené pro autentizaci a/nebo šifrování dat.

2.4.2 Popis komunikace

Klient chce komunikovat se SS. Klient se autentizuje AS, poté demonstruje TGS, že je autentizován vůči přijetí lístku pro službu (a přijme jej). Poté klient prokáže SS, že má schválení k přijetí služby.

Detailně:

1. Uživatel na klientovi zadá uživatelské jméno a heslo.
2. Klient na vložených údajích provede jednocestnou hashovací funkci. Výsledkem je klientův tajný klíč.
3. Klient zašle AS zprávu požadující službu v zastoupení uživatele (zpráva je v otevřené podobě). Např: „Uživatel XY požaduje službu“. Ani tajný klíč ani heslo se nezasílá.
4. AS ověří, zda je klient v databázi. Pokud je, AS mu zašle zpět následující dvě zprávy:
    - **Zpráva A: Klient/TGS session klíč zašifrovaný za použití tajného klíče uživatele**

[1] Kde je také vyvíjena jeho implementace

- **Zpráva B: Ticket-Granting Ticket (obsahující ID klienta, klientovu síťovou adresu, interval platnosti lístku, Klient/TGS session klíč) zašifrovaný za použití tajného klíče TGS**

5. Jakmile klient přijme zprávy A a B, dešifruje zprávu A, aby získal Klient/TGS session klíč. Tento session klíč se používá pro šifrování další komunikace s TGS.

   Klient nemůže rozšifrovat zprávu B, protože je zašifrována tajným klíčem TGS. V tomto okamžiku má klient dostatek informací, aby se mohl autentizovat vůči TGS.

6. Při požadavku na službu zasílá klient následující dvě zprávy TGS:

   - **Zpráva C: Ticket-Granting Ticket ze zprávy B a identifikátor ID požadované služby.**
   - **Zpráva D: Autentizátor (složený z klientova ID a časového razítka) zašiforvaný za použití client/TGS session klíče.**

7. TGS přijme zprávy C a D. TGS rozšifruje zprávu D (autentizátor) za použití Klient/TGS session klíče. TGS zašle následující 2 zprávy klientovi:

   - **Zpráva E: Client-to-server lístek (obsahující ID klienta, klientovu síťovou adresu, interval platnosti lístku, klient/server session klíč) zašifrovaný za použití tajného klíče service serveru (SS).**
   - **Message F: Client/server session klíč zašifrovaný client/TGS session klíčem.**

8. Po přijetí zpráv E a F od TGS má klient dostatek informací na to, aby se mohl autentizovat vůči SS. Klient zašle SS následující zprávy:

   - **Zpráva G: the client-to-server lístek zašiforvaný za použití tajného klíče SS.**
   - **Zpráva H: nový autentizátor, který obsahuje klientovo ID a časové razítko. Autentizátor je zašifrován za použití klient/server session klíče.**

9. Server dešifruje lístek za použití svého tajného klíče a zašle klientovi následující zprávu. Tím potvrzuje klientovu pravou identitu a ochotu mu sloužit.

   - **Zpráva I: (časové razítko nalezené v klientově autentizátoru + 1) zašifrováno za použití klient/server session klíče.**

10. Klient dešifruje informace a ověří správnost časového razítka. Pokud je v pořádku, může klient důvěřovat serveru a začít zasílat požadavky.

11. Server poskytuje požadované služby klientovi.

Kapitola 3

Mechanismy webové centrální autentizace

Na internetu je možno nalézt velké množství open source mechanismů pro implementaci centrální webové autentizace, využívajících cookies. V následujících sekcích jsou popsány některé zajímavé implementace centrální webové autentizace. Některé z níže popsaných mechanismů rozšiřují funkčnost SSO modelu, jiné jej splňují jen částečně.

Protože je mechanismus Cosign využíván na FIT a tato diplomová se jím bude dále zabývat, je probrán podrobněji. Většina mechanismů vychází z architektury protokolu Kerberos, proto je u popisu jejich autentizačního postupu použita podobná terminologie jako u protokolu Kerberos. Protokol autentizace u většiny webových SSO řešení je však zjednodušen, a to hlavně díky použití vrstvy SSL. Objekty, vzájemně komunikující při autentizaci, tedy nemusí explicitně zajišťovat šifrování odesílaných informací.

3.1 JA-SIG

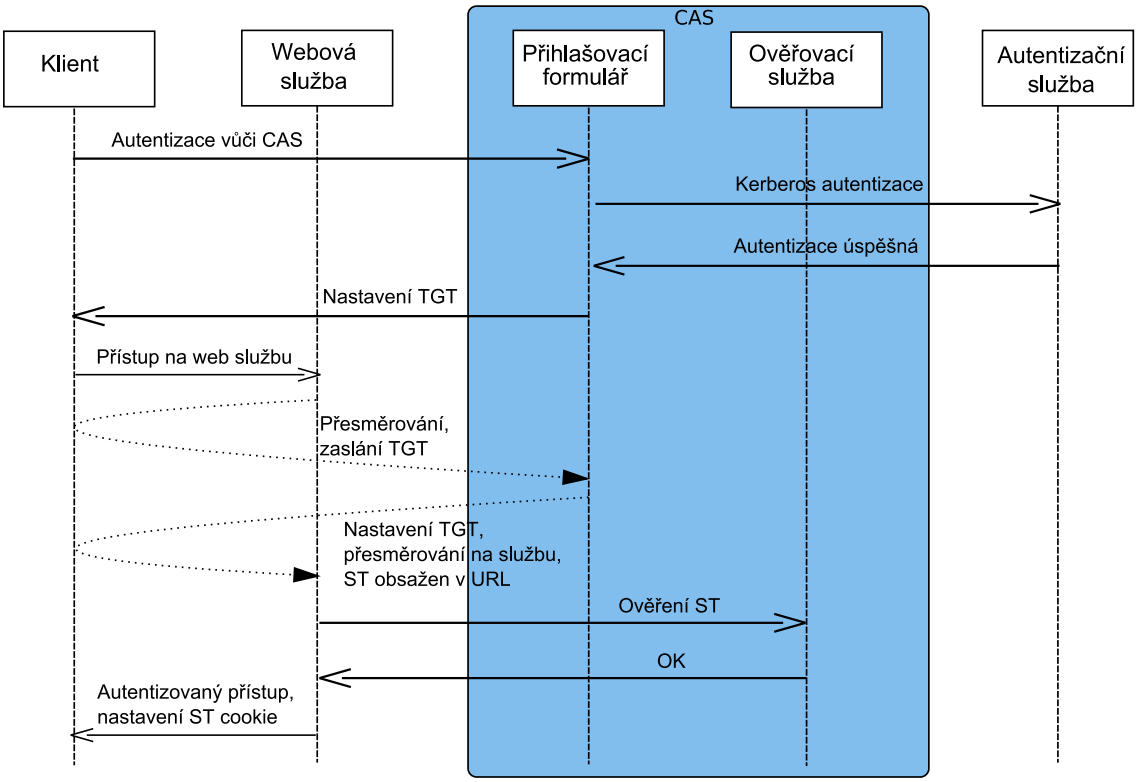
Detailní popis JA-SIG specifikace lze nalézt na domovských stránkách projektu[1].

3.1.1 Autentizační postup

Aplikace přijímající požadavek přesměruje uživatele na přihlašovací URL JA-SIG, přičemž je k dotazovacímu řetězci připojena informace identifikující "službu" - URL, na které je při úspěšném přihlášení uživatel přesměrován. Uživatel je poté vyzván k zadání uživatelského jména a hesla, které mohou poté být autentizovány různými způsoby (viz. podsekce 3.1.2). Při úspěšné autentizaci je vygenerováno Ticket Granting Cookie (TGC) - náhodný řetězec čísel, který asociuje toto TGC s uživatelem. TGC je použito jako index, podle kterého se přistupuje k vyrovnávací paměti JA-SIG serveru uchovávající ověřené identity uživatelů.

Pokud podporuje prohlížeč cookies, je mu zasláno TGC identifikující úspěšně přihlášeného uživatele. JA-SIG poté přesměruje uživatelův prohlížeč na URL dříve požadované služby a připojí Service Ticket (ST), což je další řetězec náhodných znaků asociující uživatele a požadovanou službu. Služba nyní může použít ST a ověřit jej pomocí JA-SIG ověřovací URL vložením ST a identifikátoru služby jako parametrů. Úspěšné ověření se projeví zrušením ST a vrácením uživatelského jména aplikaci. Uživatel nyní může získat TGC přímo, voláním přihlašovací URL JA-SIG. TGC může být odstraněno voláním odhlašovací URL JA-SIG, pokud tak není učiněno, implicitně vyprší jeho platnost po osmi hodinách.

Možné průběhy autentizačního procesu jsou zobrazeny na obrázcích 3.1 a 3.2



Obrázek 3.1: JA-SIG: Příklad komunikačního toku při přístupu na CAS chráněný zdoroj, při němž se klient nejdříve autentizuje.

3.1.2 Metody autentizace uživatele

Dodávaný kód obsahuje pouze základní metodu autentizace, která autentizuje uživatele, pokud je uživatelské jméno stejné jako heslo.

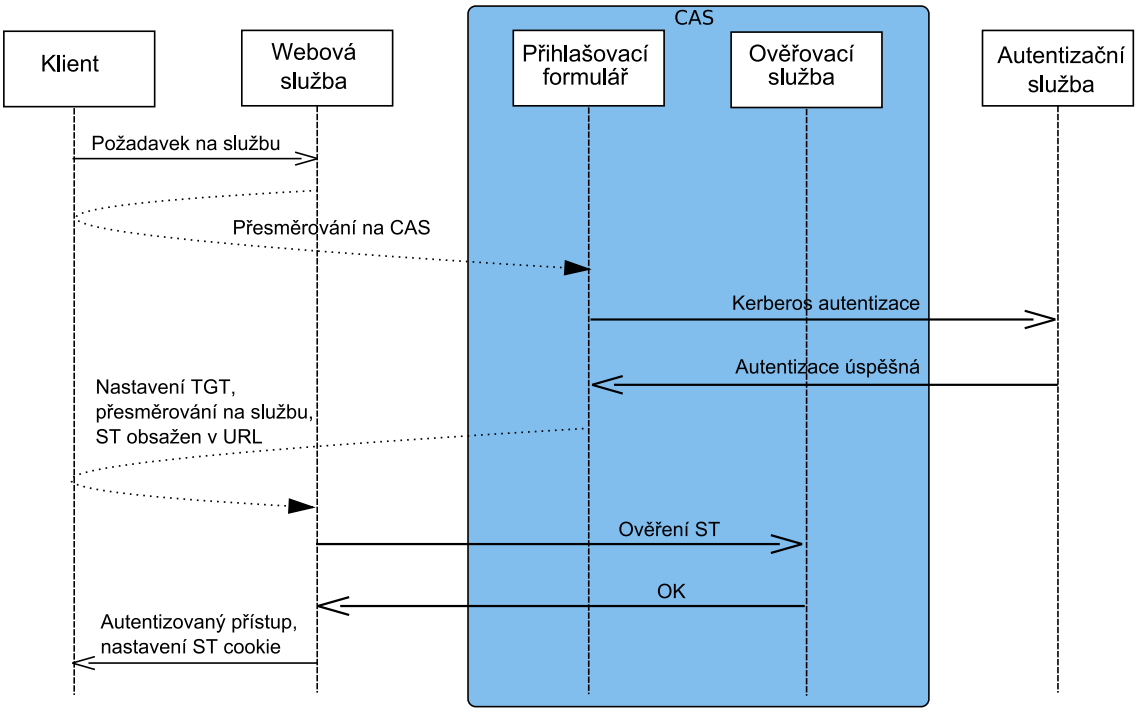
3.2 Pubcookie

Detailní popis Pubcookie specifikace lze nalézt na domovských stránkách projektu[7].

3.2.1 Model autentizace

Z názvu je patrné, že jde opět o model založený na cookies. Pubcookie přidává ustavovací fázi mezi login serverem a každým aplikačním serverem, ve které se vyjednává sdílený symetrický klíč pro zabezpečení cookies.

Uživatel požaduje po webovém serveru zdroj (pomocí URL), který je chráněný pomocí Pubcookie. Modul serveru Apache `mod_pubcookie` (nebo IIS filtr) zachytí požadavek, a pokud ještě není sezení autentizováno a nenese informace od přihlašovacího serveru (Granting Cookie), `mod_pubcookie` vygeneruje PreSession cookie určené pro aplikaci a Granting Request cookie určené přihlašovacímu serveru. Poté je prohlížeč přesměrován na adresu přihlašovacího serveru. Uživatel je poté vyzván k zadání přihlašovacích informací. Při úspěšném přihlášení je vygenerováno Granting Cookie, obsahující autentizované uživatelské jméno, a Login Cookie, pro použití při následujících požadavcích na chraněné zdroje. Poté



Obrázek 3.2: JA-SIG: Příklad komunikačního toku při neautentizovaném přístupu na CAS chráněný zdroj.

je prohlížeč přesměrován k původnímu požadovanému zdroji, požadavek nyní obsahuje Granting a PreSession Cookie. Modul `mod_pubcookie` opět zachytí požadavek, ovšem nyní nachází Granting Cookie, které ověří s PreSession Cookie. Pokud je ověření úspěšné, pak modul přiloží k původnímu požadavku uživatelské jméno a vygenerue Session Cookie pro použití při následujících požadavcích.

3.2.2 Metody autentizace uživatele

Pubcookie podporuje autentizaci skrz LDAP, shadow soubor `.htpasswd` a Kerberos.

3.3 WebAuth

Detailní popis WebAuth specifikace lze nalézt na domovských stránkách projektu[4].

3.3.1 Autentizační postup

Model WebAuth vyžaduje, aby WebAuth centrální autentizační služba (WebKDC) sdílela s každým WebAuth aplikačním serverem (WAS) sdílený klíč k zašifrování nebo dešifrování zaslaných tokenů. K zavedení a distribuci klíčů je využito Kerberosu a SSL.

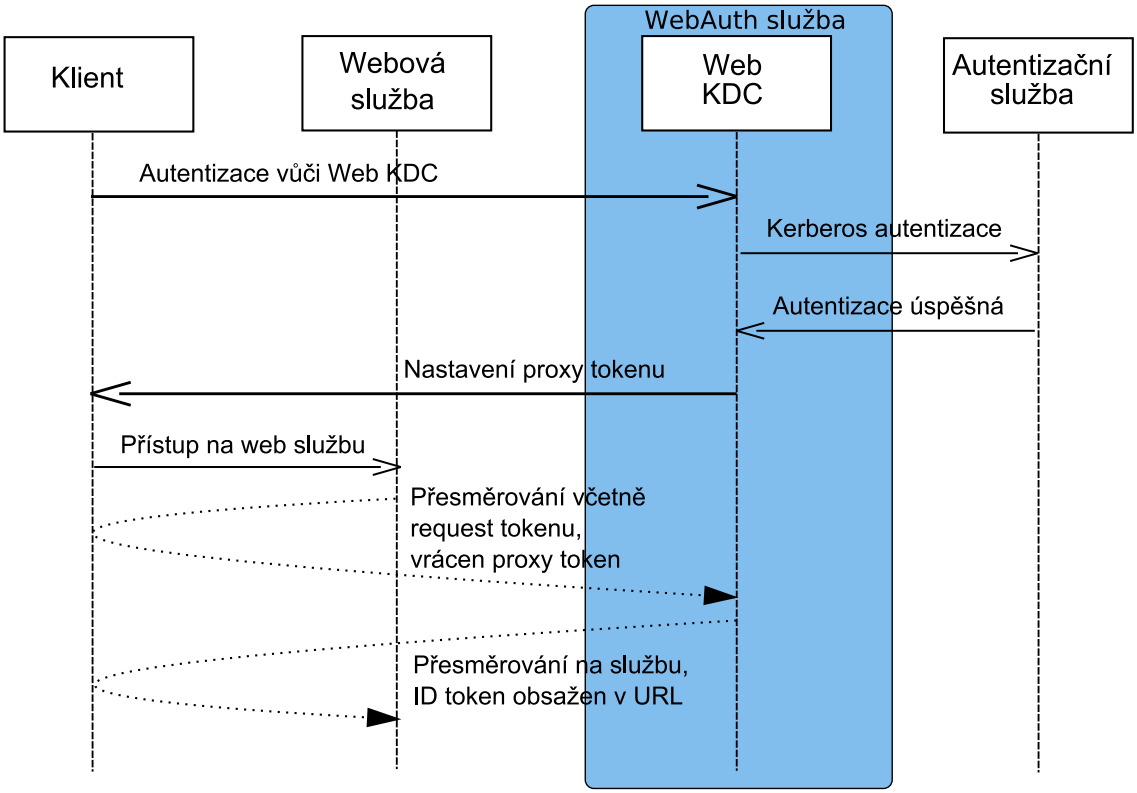
WebAuth používá cookies k uložení informací o autentizovaném uživateli. Dále WebAuth používá ID a proxy tokeny, které zpracovává WAS. Protože jsou všechny požadované údaje uloženy v cookies, WAS nemusí komunikovat s WebKDC, aby ověřil cookie a získal údaje o uživateli.

Všechny stavové infomace pro WebKDC a WAS jsou uloženy v prohlížeči uživatele ve formě cookies.

Uživatel zasílá požadavek aplikačnímu serveru WAS o zdroj (URL). Pokud ještě nebyl uživatel identifikován (zjištěno vlastnictvím WAS cookie uživatelem), je požadavek přesměrován WebAuth centru pro distribuci klíčů (WebKDC), které prohlížeči nabídne přihlašovací stránku. Po úspěšné autentizaci vygeneruje WebKDC dva tokeny. Jeden je umístěn v cookie určeném pro WebKDC a je použit pro poskytnutí jednotného přihlášení při budoucích požadavcích, druhý se zasílá zpět aplikačnímu serveru WAS, který jej při přijetí ověří.

Pokud už byl uživatel úspěšně autentizován, je povolen přístup k chráněnému zdroji.

Možné průběhy autentizačního procesu jsou zobrazeny na obrázcích 3.3 a 3.4



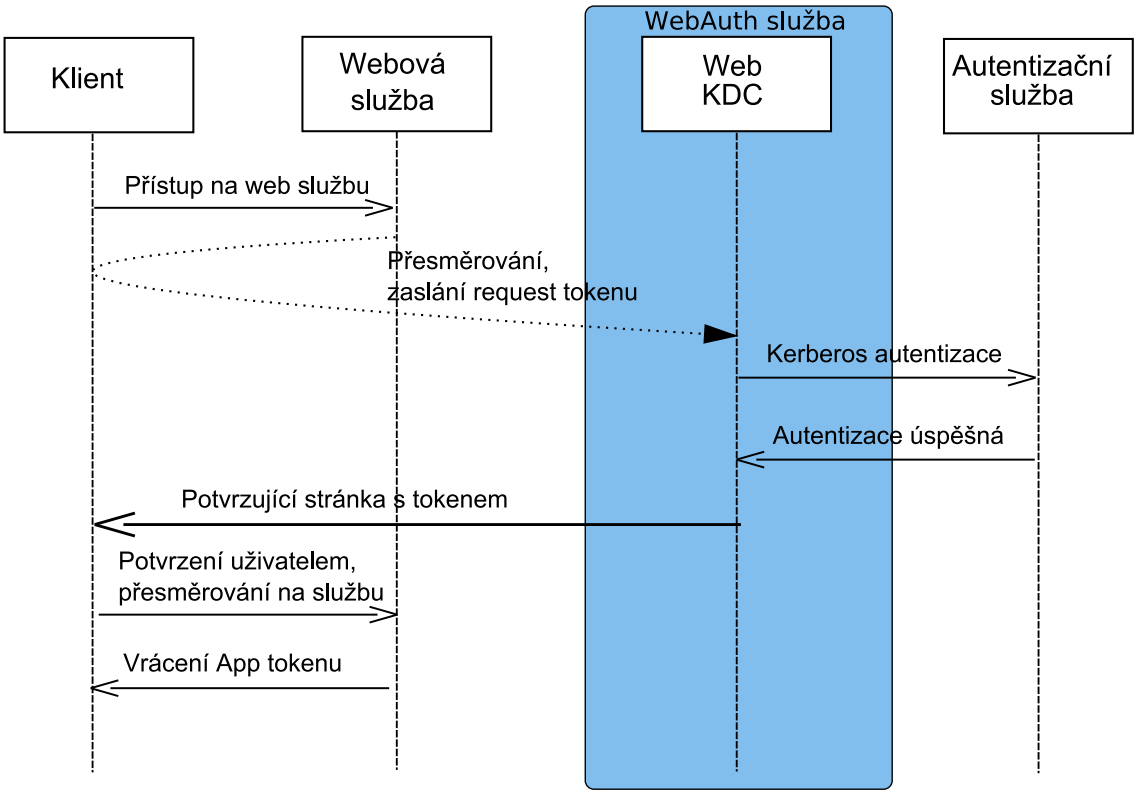
Obrázek 3.3: WebAuth: Příklad komunikačního toku při přístupu na WebAuth chráněný zdroj, při němž se klient nejdříve autentizuje.

3.3.2 Metody autentizace uživatele

WebAuth podporuje pouze Kerberos autentizaci.

3.4 Windows Live ID Web Authentication

Jednotná přihlašovací služba vyvíjenou a nabízená společností Microsoft, která umožňuje přihlášení k více webovým stránkám za použití jednoho účtu. Je součástí Windows Live



Obrázek 3.4: WebAuth: Příklad komunikačního toku při neautentizovaném přístupu na WebAuth chráněný zdroj.

ID balíku, který nabízí také autentizaci jiných než webových služeb. Live ID je prezentováno jako SSO řešení, při testování ovšem většina služeb vyžadovala opětovné zadávání uživatelského jména či hesla. Live ID se také odlišuje tím, že uživatel není v případě nutnosti automaticky přesměrován na přihlašovací server, ale dostane se k němu až po kliknutí na příslušný odkaz.

3.4.1 Autentizační postup

Postup při přihlašování je takovýto:

1. Uživatel používající webový prohlížeč navštíví poprvé zabezpečenou webovou stránku, přičemž není přihlášen k systému Live ID.
2. Služba vrátí uživateli stránku, která v IFRAME elementu zobrazí speciální přihlašovací odkaz.
3. Uživatel na odkaz klikne.
4. Uživateli je nabídnuta přihlašovací stránka Windows Live ID. Uživatel vyplní své přihlašovací údaje (e-mail a heslo) a potvrdí formulář.
5. Windows Live ID autentizační server přijme přihlašovací požadavek a ověří zaslané přihlašovací údaje.

6. Pokud jsou přihlašovací údaje platné, autentizační server odpoví přesměrování uživatele na původně požadovaný zdroj. Zároveň je uživateli poskytnut token ve formě POST parametru. Tento token slouží jako potvrzení, že Windows Live ID ověřilo identitu uživatele. Zabezpečený zdroj rozšifruje token a získá tím unikátní identifikátor uživatele.

7. Uživateli je zobrazen požadovaný zabezpečený zdroj.

3.5 Cosign - podrobný popis

Open source webový single sign-on mechanismus. Obsahuje následující součástí:

daemon - centrální autorita, která udržuje informaci o vytvořených sezeních

CGI web login - CGI[1] skript poskytující webové uživatelské rozhraní k autentizaci, nejčastěji formou přihlašovacího formuláře

filtr - implemenetován jako modul do web serveru, který zajišťuje, že k určitým zdrojům přistupují pouze autentizovaní uživatelé

Web login aplikace a jednotlivé filtry, které řídí přístup k zdrojům (aplikacím, datům,. . .) komunikují s daemonem prostřednictvím zabezpečeného SSL kanálu, a proto každá součást potřebuje certifikát podepsaný platnou certifikační autoritou.

3.5.1 Autentizační postup

Proces autentizace začne, když uživatel přistupuje k oblasti na webovém serveru, pro kterou je nakonfigurován filter, aby vyžadoval autentizaci. Filter zjišťuje, zda je v uživatelově požadavku přítomno platné service cookie, není-li přítomno, vystaví nové a přesměruje uživatelův prohlížeč na Cosign CGI, zároveň s vložením service cookie a návratového URL do dotazovacího řetězce.

Cosign CGI zjišťuje, zda je přítomno platné login cookie. Jestliže je přítomno, CGI registruje nové service cookie s existujícím login cookie a přesměruje uživatelův prohlížeč na návratovou URL adresu. Jestliže není login cookie přítomno, CGI uživateli nabídne autentizační formulář a po následné úspěšné autentizaci vytvoří login cookie, zaregistruje s ním service cookie a přesměruje uživatelův prohlížeč na návratovou adresu.

Když se uživatelův prohlížeč opět pokusí přistoupit k filtrem chráněné aplikaci, filter opět zkontroluje přítomnost service cookie. Je-li service cookie v lokální databázi filtru platné, nebo ověří-li platnost service cookie cosign démon, může filtr vytvořit prostředí s autentizačními informacemi uživatele a povolí přístup ke chráněnému zdroji.

Možné průběhy autentizačního procesu jsou zobrazeny na obrázcích 3.5 a 3.6. Podrobný diagram průběhu komunikace je uveden v příloze A.

3.5.2 Použitá cookies

Cosign používá dvě různé třídy cookies - login cookie a service cookie. Obojí cookies jsou hostitelská cookie, tzn. že pouze server, který je vystavil, má právo k jejich získání. Nové hodnoty cookies mají tvar 128 náhodných znaků. Tyto náhodné řetězce jsou generovány za použití knihovny OpenSSL. V řetězci mohou být použity znaky malých i velkých písmen, číslic, '+' a '-'. Znak "/" není povolen, protože jej Cosign používá jako oddělovač.

[1] Common Gateway Interface

Login cookie - klíče generované CGI, podle kterých se daný prohlížeč identifikuje cosign serveru. Jejich formát je následující:

`cosign=[128 náhodných znaků]/čas_vytvoření_cookie/registrační_počet`

Čas vytvoření se používá k zajištění obnovení platnosti login cookie, které mohou být některými prohlížeči zadržené. Pokud je login cookie starší než 24 hodin (lze nastavit v konfiguračním souboru), je vydáno nové cookie. Registrační počet se používá k detekci registračních smyček. Pokud navštíví uživatel přihlašovací stránku více než udává konstanta `MAXLOOPCOUNT` během `LOOPWINDOW` sekund, je jeho prohlížeč přesměrován na stránku s přerušením cyklu, kde jsou popsány informace o problému se zaregistrováním. Tato situace může nastat např. v případě chyby na straně filtru.

Service cookie - klíče, podle kterých se daný prohlížeč identifikuje aplikačnímu serveru, jsou generovány filtrem. Jejich formát je následující:

`cosign-jméno_služby=[128 náhodných znaků]/čas_vytvoření_cookie`

Při vytváření nového cookie připojí filtr před jméno služby předponu `cosign-`. Stejně jako u login cookie, i u service cookie se čas vytvoření používá k zajištění obnovení platnosti service cookie, které mohou být některými prohlížeči zadržené. Pokud je service cookie starší než čas vypršení, nastavený podle konfigurace filtru, je vydáno nové cookie.

3.5.3 Filtr

Filtr je umístěn na aplikačním serveru a není součástí centralizované infrastruktury Cosign. Určuje které oblasti webové prezentace jsou chráněny mechanismem cosign a které ne. Pokud se uživatel pokusí přistoupit k chráněné oblasti, filtr zajistí, že je uživatel autentizován, získá jeho uživatelské jméno, autentizační doménu, IP adresu a volitelně i Kerberos lístek. Tyto informace předá chráněným aplikacím, kde mohou být dále zpracovány například autorizačními mechanismy.

3.5.4 Dotazovací řetězec

Dotazovací řetězec se používá k zasílání informací mezi filtrem a CGI. Též je využíván CGI k zasílání informací mezi sebou. Dále je popsán dotazovací řetězec uživatelského prohlížeče a jeho přesměrování.

Dotazovací řetězec používaný filtrem

Informace zaslaná CGI pomocí dotazovacího řetězce se skládá z registrační URL adresy, která je specifikovaná v konfiguraci filtru, otazníku oddělujícího data dotazovacího řetězce, seznamu požadovaných faktorů (pokud jsou některé požadované), uživatelova service cookie a URL adresy, na kterou se prohlížeč přesměruje, pokude je fáze REGISTER úspěšně hotova. Tato informace má tedy následující formát:

`register-url?[factors=factor1[,factor2]...&]service=cookie[;]&reffering-url`

Dotazovací řetězec používaný CGI

CGI si kromě toho, co dostane od filtru, může zaslat ještě dodatečný obsah. Klíčové slovo `basic` si může CGI volitelně zaslat, aby zjistilo, zda jsou v prohlížeči povolena cookies. Navíc `basic` znamená, že CGI pracuje v režimu kompatibilním s BasicAuth a je nastavena proměnná prostředí `REMOTE_USER`.

```
register-url?[basic&][[factors=factor1][,factor2]...&]
service=cookie[;]&reffering-url
\begin{Verbatim}

\subsection{Skripty CGI}
Cosign má 2 CGI - cosign.cgi a logout
\begin{description}
\item[cosign.cgi] - ''Logovací''' CGI je zodpovědné za logování uživatelů do centrálníh

CGI také umožňuje vyzvat uživatele k opětovnému zadání jeho hesla za účelem přístupu ke

\item[logout] - toto CGI je zodpovědné za odhlašování uživatelů z centrálního cosign se
Z důvodu implementace cacheování dat ve filtrech, poslední otevřená aplikace bude uživa
\end{description}

\subsection{Démoni}
Démoni, neboli procesy běžící na pozadí. Cosign je tvořen dvěma démony - cosignd a mons

\begin{description}
\item[Cosignd] - centrální démon, serverová implementace cosign protokolu (popsán dále)

\item[Monster] - slouží dvěma různým účelům. Ve všech případech je to mechanismus, kter
\end{description}

\subsection{Lokální úložiště cookie}
Démoni a filtry užívají následující formát pro lokální ukládání cookie:

Filtr:
\begin{Verbatim}[frame=single, fontsize=\footnotesize]
i        ip adresa
p        principal (uživatelské jméno)
r        realm (doména, první faktor)
f        všechny faktory
k        cesta k lístku kerberos (volitelné)
```

Například:

```
i66.93.92.59
ptestuser
rUMICH.EDU
fUMICH.EDU mtoken
k/ticket/EjAP2EqDy-5
```

Démon:

```
v        verze
s        stav
i        počáteční ip adresa
j        další ip adresa
```

```
p       principal (uživatelské jméno)
r       faktor(y)
t       čas, kdy byly data uloženy (v sekundách od roku 1970)
k       cesta k lístku kerberos (volitelné)
```

3.5.5 Protokol

Cosignd podporuje následující požadavky protokolu:

QUIT - Ukončení sezení

NOOP - Prázdná operace

HELP - Zobrazení pomocné zprávy

STARTTLS - Začátek komunikace TLS. Tento příkaz musí být zaslán dříve, než klient zašle některý z příkazů LOGIN, REGISTER, CHECK, LOGOUT, RETRIEVE, DAEMON nebo TIME. Příkazy QUIT, NOOP a HELP jsou jediné povolené, dokud není STARTTLS voláno.

LOGIN - Asociace login cookie s uživatelem, faktory, IP adresou a časovým razítkem. Pokud jej CGI použije, je uživatel úspěšně autentizován.

REGISTER - Asociace service cookie s login cookie. Tento příkaz používá CGI.

LOGOUT - Zneplatnění uživatelova login cookie. Jakýkoliv další CHECK příkaz vrací status "odhlášen". Tento příkaz používá CGI.

CHECK - Získání informací o uživateli založené na hodnotě service cookie předané cosignd. Tento příkaz je používán jak CGI, tak filtry (službami).

RETRIEVE - Získání Kerberos ticketu nebo proxy cookies pro vícevrstvou autentizaci.

DAEMON - Replikace, zamezuje serveru, aby replikoval sám sebe. Tento příkaz je užíván démony cosignd a monster.

TIME - Propagace informace o časovém razítku a stavu login cookies. Tento příkaz je užíván démonem monster.

3.5.6 Specifikace autentizačního filtru

Vrstva modulu

Při přijetí požadavku od klienta zjišťuje autentizační filtr přítomnost platného service cookie. Je-li cookie prezentované uživatelem shodné s cookie v lokální cookie databázi filtru, ověří filtr jeho časové razítko. Pokud je časové razítko stále platné, povolí filtr přístup uživateli, je-li již časové razítko expirované, filtr začne komunikaci s cosignd démonem, aby uživatele ověřil. Po této komunikaci může cosignd démon odpovědět sedmi možnými stavy:

Cookie valid - prezentované service cookie je platné, IP adresa je správná (pouze pokud kontrola IP adresy povolena, nebude fungovat s firewally/NATy), uživatelské jméno souhlasí s uživatelským jménem asociovaným s cookie, uživatel není odhlášen.

No cookie - uživatel neprezentoval žádné cookie pro tuto službu

Bad cookie - prezentované service cookie není v databázi service cookies

Wrong IP - uživatelova ip adresa nesouhlasí s IP adresou asociovanou s tímto cookie

Wrong user - uživatelské jméno neshouhlasí s uživatelským jménem asociovaným s tímto cookie

Logged out - uživatel je odhlášen od centrálního Cosign serveru

Unknown - nastala neočekávaná chyba

Odpovědí na první stav "cookie valid" je aktualizace časového pole v cookie databázi a potvrzení platnosti cookie zasláním zprávy DECLINED (znamenající, že filtr odmítá pokračovat v požadavku, protože všechny kontroly souhlasí). Odpověď DECLINED způsobí, že požadavek na zdroj je zaslán obvyklým způsobem a zdroj je normálně vrácen. Předtím, než je vrácena odpověď DECLINED, musí modul filtru zpřístupnit následující proměnné prostředí zbytku smyčky požadavek/odpověď:

- AUTHTYPE: "Cosign"
- COSIGN SERVICE: jméno cosign služby která řídí tuto lokaci nebo adresář
- REMOTE USER: uživatelské jméno autentizovaného uživatele
- COSIGN FACTOR: všechny faktory, kterým autentizovaný uživatel vyhovuje
- REMOTE REALM: (pro zpětnou kompatibilitu) uživatelův první autentizační faktor
- KRB5CCNAME: pokud se používá Kerberos, cesta ke Kerberos ticketu

V odpovědi na zbývajících šest cookie stavů generuje modul filtru nové service cookie, zasílá set-cookie hlavičku se service cookie a vrací REDIRECT. Uživatel je přesměrován na centrální CosignRedirect URL (nastaveno v konfiguraci) se service cookie připojeným k dotazovacímu řetězci.

Vrstva datového úložiště

Databáze cookies je implementována jako část UNIXového souborového systému. Záznam je vlastně obyčejný soubor. Jeho vyhledávací klíč (jméno souboru) je náhodná hodnota vygenerovaná podle jména uživatelova service cookie. Data uložená v každém záznamu mají tento formát:

Username - uživatelské jméno autentizovaného uživatele

Realm - první faktor se kterým se uživatel autentizoval

Factor - všechny faktory se kterými se uživatel autentizoval

IP address - IP adresa uživatelova prohlížeče

K5 ticket - pokud existuje, tak cesta ke Kerberos 5 lístku

Time - čas naposledy použitý při validaci (uložen v mtime[2] hodnotě souboru)

[2] Časové razítko posledního zápisu do souboru

Síťová vrstva

Síťová vrstva modulu je zodpovědná za ustavení a správu SSL šifrovaných spojení s centrálním cosignd serverem, uchování stavu spojení, implementaci zabezpečení výpadku, rozdělení zátěže a vzájemnou interakci s cosignd démonem.

Protokol

Filtr smí užívat pouze tyto (dříve popsané) direktivy protokolu:

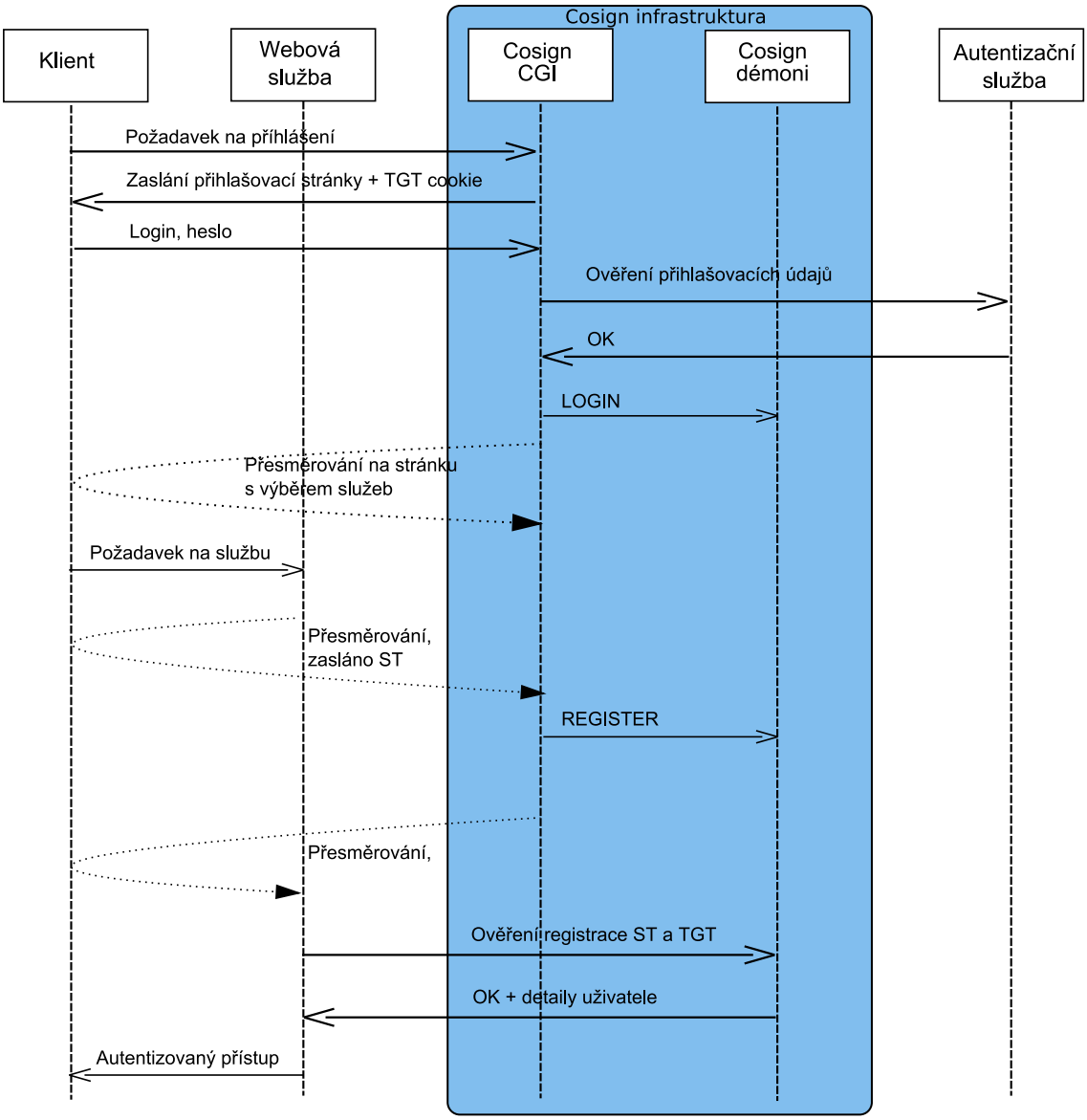
- CHECK `service_cookie`
- RETR `service_cookie` "tgt"/"cookies"'
- STARTTLS [2]
- NOOP
- QUIT

Konfigurace prostředí modulu

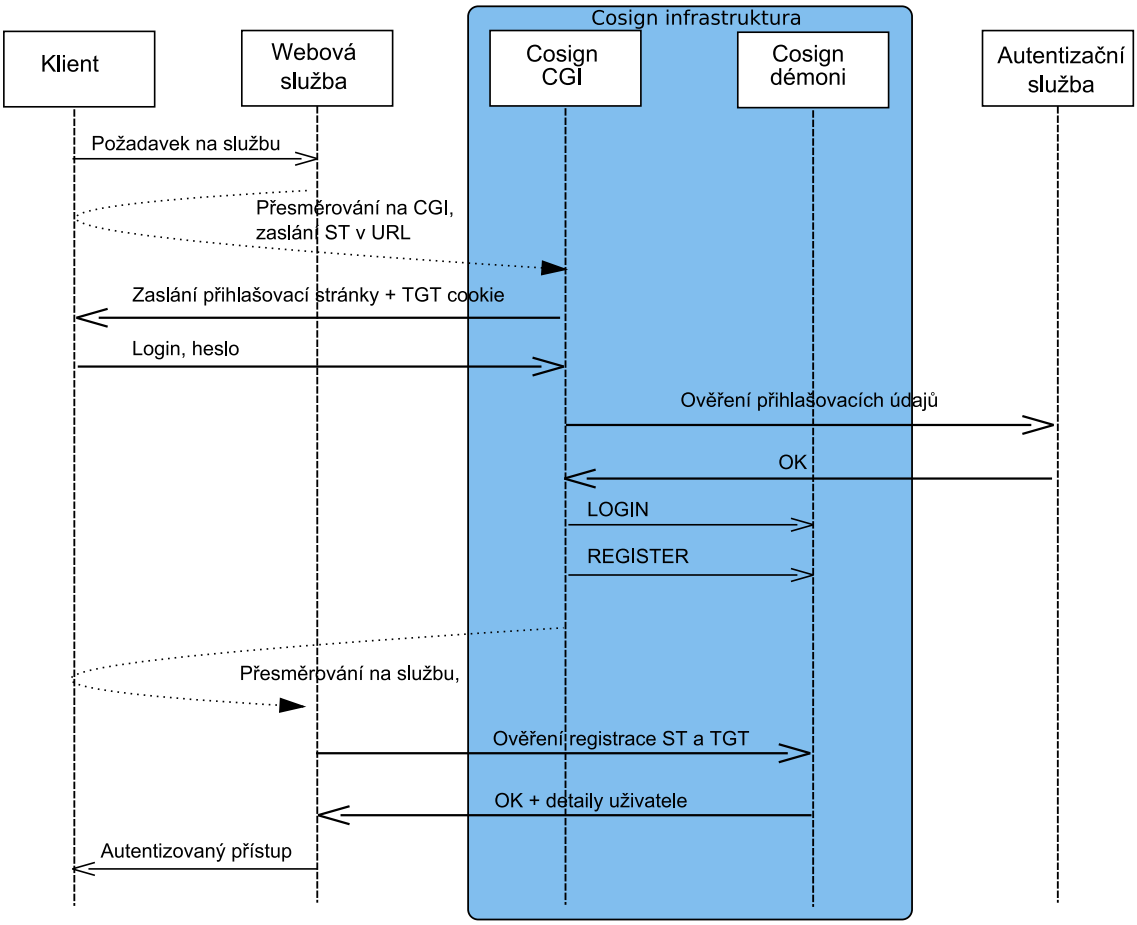
Konfigurační direktivy rozeznávané modulem a jejich význam jsou uvedeny v příloze B.

3.6 Srovnání vybraných systémů

Srovnávané systémy se většinou v hlavních rysech velmi neliší (kromě Live ID), naší potřebě však nejvíce vyhovuje systém Cosign. Díky přehlednému komunikačnímu protokolu, zabezpečení komunikace SSL protokolem a výměně identifikačních tokenů namísto přihlašovacích údajů je bezpečným a přitom přehledným systémem. Oproti ostatním implementacím nabízí možnost instalace filtru v prostředí serveru Apache, IIS a JavaServletů. Při použití komunikačního protokolu verze 2 poskytuje také možnost použití více autentizačních mechanismů zároveň a implemntaci vlastního způsobu autentizace. Výhodou pro Cosign je již zmíněná specifikace pro vývoj vlastního filtru.



Obrázek 3.5: Cosign: Příklad komunikačního toku při přístupu na Cosign chráněný zdroj, při němž se klient nejdříve autentizuje.



Obrázek 3.6: Cosign: Příklad komunikačního toku při neautentizovaném přístupu na Cosign chráněný zdroj.

Kapitola 4

Z pohledu bezpečnosti

Jelikož je Cosign systémem zajišťujícím bezpečnou vícenásobnou autentizaci, je zřejmé, že musí používat i bezpečné mechanismy, aby zajistil důvěrnost a integritu přenášených informací. Jak pro přenos přihlašovacích informací od uživatele k web loginu, tak pro komunikaci filtru s CGI, či komunikaci CGI s jinými CGI, je využíváno moderní kryptografie. Detailní popis kryptografických algoritmů čerpán z [15] a [14].

4.1 Kryptografické zabezpečení Cosign komunikace

Před započetím jakékoliv komunikace musí být zaručeno, že zasílaná data nemohou být nikým odposlechnuta, změněna, či podvržena. Aby nedošlo k některému z těchto bezpečnostních rizik, komunikace Cosign probíhá podle Transport Layer Security (TLS) protokolu, založeném na asymetrické a symetrické kryptografii.

4.2 Symetrická kryptografie

Algoritmy symetrické kryptografie jsou třídou algoritmů, které užívají identické kryptografické klíče jak pro zašifrování, tak pro dešifrování.

Šifrovací klíč souvisí s dešifrovacím klíčem tak, že jsou identické, nebo existuje jednoduchá transformace, díky které získáme z jednoho klíče klíč druhý. Klíč v praxi reprezentuje sdílené heslo mezi dvěma nebo více stranami, které se používá ke správě zabezpečeného informačního spojení.

4.2.1 Typy algoritmů pro symetrickou kryptografii

Algoritmy symetrické kryptografie mohou se dělí na proudové šifry nebo blokové šifry. Proudové šifry zpracovávají otevřený text po jednotlivých bitech. Blokové šifry rozdělí otevřený text na bloky stejné velikosti a doplní vhodným způsobem poslední blok na stejnou velikost. Poté šifrují každý blok jako samostatnou jednotku. Běžně se používají bloky o velikosti 64 bitů a větší. Mezi populární a v praxi používané symetrické algoritmy patří například AES (aka Rijndael), Blowfish, RC4, TripleDES a IDEA.

4.3 Asymetrická kryptografie

Známá také pod názvem kryptografie veřejného klíče, je formou kryptografie, ve které uživatel vlastní pár kryptografických klíčů - veřejný klíč a soukromý klíč. Privátní klíč je tajný, zatímco veřejný klíč může být volně šířen. Oba klíče jsou matematicky spojeny, přičemž soukromý klíč nemůže být z veřejného získán. Zpráva zašifrovaná veřejným klíčem může být rozšifrována pouze za použití odpovídajícího soukromého klíče.

Dvěma hlavními odvětvími asymetrické kryptografie jsou:

Šifrování veřejným klíčem - Zpráva zašifrovaná veřejným klíčem adresáta nemůže být rozšifrována nikým jiným, kromě adresáta vlastnícího odpovídající privátní klíč. Tímto se zajišťuje dosažení důvěrnosti.

Digitální podpisy - Zpráva podepsaná soukromým klíčem odesilatele může být ověřena pouze někým, kdo má přístup k veřejnému klíči odesilatele, tzn. lze prokázat, že zprávu opravdu napsal odesilatel a nebyla zfalšovaná. Tímto je zajištěna integrita, autentizace a nepopiratelnost.

Problémem pro asymetrickou kryptografii je prokázání, že veřejný klíč je autentický a nebyl změněn nebo podvržen nebezpečnou třetí stranou. Obvyklým přístupem je použití public-key infrastruktury (PKI), ve které je více třetích stran, známých jako certifikační autority (CA), které certifikují vlastnictví párů klíčů. Jiný přístup, užitý technologií PGP, je metoda "web of trust" zajišťující taktéž autentičnost klíčů v páru.

Techniky využívající kryptografii veřejným klíčem jsou technologicky náročnější, než čistě symetrické algoritmy. Rozumné užití těchto technik umožňuje širokou škálu použití. V praxi se z výkonnostních důvodů asymetrická kryptografie užívá v kombinaci se symetrickou kryptografií.

Asymetrické šifrování se nepoužívá pro zabezpečený přenos velkých objemů dat. Pro jejich šifrování je použito tajného sdíleného klíče, který ovšem musí znát obě strany. Pro distribuci tohoto tajného klíče je použito právě asymetrické kryptografie, a to takovýmto způsobem:

1. Strana A vygeneruje náhodný tajný sdílený klíč a zašifruje jej veřejným klíčem strany B.
2. Pak takto zašifrovaný klíč odešle straně B.
3. Pouze a jedině strana B za použití svého soukromého klíče může tuto zašifrovanou zprávu rozšifrovat.
4. Ke komunikaci se pak používá zmíněný tajný sdílený klíč, který znají obě komunikující strany.

Pro účely digitálního podpisu odesilatel aplikuje na zprávu hash funkci a poté podepíše výsledný hash (velikost maximálně stovky Byte). Odešle nezašifrovanou zprávu a hash zašifrovaný svým soukromým klíčem. Příjemce taktéž spočítá hash zprávy a porovná tento hash s hodnotou, kterou získá rozšifrováním pomocí odesilatelova veřejného klíče. Pokud se hashe zpráv neliší, je zajištěno, že zpráva nebyla po cestě změněna.

4.4 TLS protokol

Primárním cílem TLS protokolu, formálně popsaného RFC doporučením[9], je zajištění důvěrnosti a integrity dat mezi dvěma komunikujícími aplikacem. Protokol je složen ze dvou vrstev: **TLS Record** protokolu a **TLS Handshake** protokolu. Na nejnižší vrstvě ležící nad některým spolehlivým transportním protokolem (např. TCP) se nachází TLS Record protokol. TLS Record protokol poskytuje bezpečné spojení, které má dvě hlavní vlastnosti:

- spojení je důvěrné. Pro zašifrování dat je použita symetrická kryptografie (např. DES, RC4, atd.). Klíče pro toto symetrické spojení jsou generovány unikátně pro každé spojení a jsou založeny na tajemství vyjednaném druhým protokolem (TLS Handshake). Record protokol může být též použit bez šifrování.

- spojení je spolehlivé. Zároveň s přenosem zprávy se přenáší také integritní kontrola Message Authentication Code (MAC). Pro výpočet MAC se využívají bezpečné hashovací funkce (např. SHA, MD5, ...).

TLS Record protokol se používá k zapouzdření různých protokolů vyšších vrstev. Jedním takovým zapouzdřeným protokolem může být TLS Handshake protokol, který serveru a klientu dovoluje autentizovat jeden druhého a vyjednat šifrovací algoritmus a kryptografický klíč před tím, než aplikační protokol odešle nebo přijme první Byty dat. TLS Handshake protokol poskytuje bezpečné spojení, které má tři základní vlastnosti:

- identita komunikujících stran může být autentizována za použití asymetrické kryptografie (např. RSA, DSS, ...). Tato autentizace může být volitelná, ale obvykle je vyžadována alespoň pro jednoho z komunikujících.

- dohodnutí sdíleného tajemství je bezpečné. Dohodnuté heslo je nepřístupné naslouchání. Pro kohokoliv neautentizovaného je heslo nepřístupné, stejně tak i pro útočníka, který by se umístil do centra komunikace[1].

- dohodnutí je důvěryhodné. Útočník nemůže modifikovat zprávy zasílané při dohadování hesla, aniž by byl odhalen komunikujícími.

Výhodou TLS je fakt, že je aplikačně/protokolově nezávislý. Protokoly vyšší třídy se mohou vrstvit nad TLS protokolem transparentně. TLS standard nespecifikuje techniku zajištění bezpečnosti. Rozhodnutí jak iniciovat TLS ustavení spojení a jak interpretovat výměnu autentizačních certifikátů je ponecháno designérům a implementátorům protokolů, které běží nad TLS. Protože je TLS protokol založen na specifikaci protokolu Secure Socket Layer (SSL) verze 3.0 publikovaného společností Netscape a výrazně se od něj neliší, je běžné používat pro TLS také zkratku SSL či TLS/SSL

Ustavení TLS/SSL spojení (SSL handshake) pak probíhá následovně:

1. Klient pošle serveru požadevek na SSL spojení, spolu s různými doplňujícími informacemi (verze SSL, nastavení šifrování atd.).

2. Server pošle klientovi odpověď na jeho požadavek, která obsahuje stejný typ informací a hlavně certifikát serveru.

[1] Man in the middle útok

3. Podle přijatého certifikátu si klient ověří autentičnost serveru. Certifikát také obsahuje veřejný klíč serveru.

4. Na základě dosud obdržených informací vygeneruje klient základ šifrovacího klíče, kterým se bude šifrovat následná komunikace. Ten zašifruje veřejným klíčem serveru a pošle mu ho.

5. Server použije svůj soukromý klíč k rozšifrování základu šifrovacího klíče. Z tohoto základu vygenerují jak server, tak klient hlavní šifrovací klíč.

6. Klient a server si navzájem potvrdí, že od teď bude jejich komunikace šifrovaná tímto klíčem. Fáze handshake tímto končí.

7. Je ustaveno zabezpečené spojení šifrované vygenerovaným šifrovacím klíčem.

8. Aplikace od teď dál komunikují přes šifrované spojení. Například POST požadavek na server se do této doby neodešle.

4.5 Kryptografické možnosti jazyka PHP

PHP je reflektivní[2] počítačový programovací jazyk navržený pro vytváření dynamických webových stránek. PHP se používá hlavně jako skriptovací jazyk na straně serveru, ale může být použit i v prostředí příkazové řádky nebo v samostatné grafické aplikaci. PHP je původně strukturovaný programovací jazyk, podpora pro objektově orientované programování (OOP) byla přidána do verze PHP 3. Práce s objekty byla kompletně přepsána pro verzi PHP 5, rozšiřujíce sadu objektových rysů a zlepšujíce výkon. PHP 5 představilo soukromé a chráněné členské proměnné a metody, zároveň s abstraktními třídami a abstraktními metodami. Byl do něj také zanesen standardní způsob deklarace konstruktorů a destruktorů, podobný tomu, který se používá v jiných objektově orientovaných jazycích (např. C++). Výrazným přínosem bylo přidání modelu zachytávání výjimek[16].

PHP je živým jazykem, jehož specifikace se neustále vyvíjí a modernizuje. Stejně tak je tomu i s jeho nabídkou kryptografických funkcí.

4.5.1 Podpora symetrické kryptografie - mcrypt

Knihovna mcrypt nabízí uživateli velké množství symetrických šifrovacích algoritmů. Mcrypt není standardní součástí instalace PHP, při kompilaci PHP je třeba pro aktivaci tohoto rozšíření zadat direktivu `--with-mcrypt[=DIR]`, kde `[DIR]` je instalační adresář mcrypt. Přesto je dnes mcrypt na většině webhostingových serverů standardně podporován.

Aktuální verze knihovny mcrypt[3] podporuje následující známé blokové algoritmy: CAST, LOKI97, RIJNDAEL (AES), SAFERPLUS, SERPENT a následující proudouvé šifry: ENIGMA (crypt), PANAMA, RC4 a WAKE.

4.5.2 Podpora asymetrické kryptografie - OpenSSL

Toto rozšíření PHP využívá funkce OpenSSL pro tvorbu a ověřování podpisů a pečetění (kódování) a otvírání (dekódování) dat. Samotné OpenSSL nabízí mnoho vlastností, které toto rozšíření v současnosti nepodporuje. Některé z nich mohou být v budoucnu přidány.

[2] Reflektivní programovací jazyky mohou dynamicky modifikovat strukturu programu v době běhu
[3] Mometálně verze libmcrypt-2.58

Abychom mohli používat tyto kryptografické funkce, musíme nainstalovat OpenSSL rozšíření. PHP je třeba zkompilovat s volbou `--with-openssl`. OpenSSL rozšíření je dnes na většině webhostingových serverů standardně podporováno.

Parametry s klíči/certifikáty

Několik OpenSSL funkcí potřebuje parametr s klíčem nebo certifikátem. PHP 4.0.5 a starší musí pro klíč a certifikát používat zdroj vrácený některou z funkcí `openssl_get_xxx`. Pozdější verze PHP mohou používat libovolnou z těchto metod:

Certifikáty:

- X.509 zdroj vrácený funkcí `openssl_x509_read()`
- Řetězec ve formátu `file://path/to/cert.pem`; uvedený soubor musí obsahovat PEM-zakódovaný certifikát[4]
- Řetězec obsahující PEM-zakódovaný certifikát

Veřejné a soukromé klíče:

- Zdroj klíče vrácený funkcí `openssl_get_publickey()` nebo `openssl_get_privatekey()`
- Pouze veřejné klíče: zdroj X.509
- Řetězec ve formátu `file://path/to/file.pem` - uvedený soubor musí obsahovat PEM-zakódovaný certifikát nebo veřejný klíč (může obsahovat oba)
- Řetězec obsahující obsah certifikátu/klíče, PEM-zakódovaný
- Pro soukromé klíče lze také použít syntaxi `array(\$key, \$heslo)` kde `$key` reprezentuje klíč zadaný pomocí file:// nebo textového uvedení zmíněného výše a `$heslo` reprezentuje řetězec obsahující heslo pro tento soukromý klíč

Podrobná specifikace všech kryptografických funkcí nabízených knihovnou OpenSSL je dostupná na domovských stránkác projektu PHP[6].

[4]Privacy Enhanced Mail - standar pro zabezpečení elektronické pošty za použití kryptografie veřejného klíče

Kapitola 5

Implementace Cosign filtru

5.1 Příprava Cosign infrastruktury

Abychom mohli testovat postupně se vyvíjející filtr, je třeba nejdříve nainstalovat kompletní Cosign infrastrukturu. Na stránkách projektu Cosign byla v době psaní této diplomové práce zveřejněna verze 2.0.2a Cosign infrastruktury. Z důvodu opravy některých chyb bylo později použito i verze 2.1.0RC2.[1]

5.1.1 Instalace Cosign

Serverová část Cosign je určena pro UNIXovou platformu. Filtr obsažený v balíčku je standrdně také určen pro UNIXovou distribuci serveru Apache, lze však dodatečně instalovat i filtr pro IIS nebo Javu. Příkazem pro download archivu získáme poslední verzi Cosign systému, kterou uložíme do požadovaného adresáře a dekomprimujeme:

```
# cd /opt/cosign/src
# wget http://www.umich.edu/~umweb/downloads/cosign-2.0.2a.tar.gz
# tar xvfz cosign-2.0.2a.tar.gz
```

Spustíme konfigurační skript, je však třeba poradit kompilátoru direktivu `-D_LARGEFILE64_SOURCE` pro úspěšnou kompilaci APR[2] zdrojových souborů. Přednastavíme cílový adresář pro instalaci, umístění konfiguračního souboru, databáze Apache filtru, databáze serveru, umístění adresáře certifikační autority a certifikátů:

```
# CPPFLAGS="-D_LARGEFILE64_SOURCE" ./configure \
--enable-apache2=/usr/local/apache2/bin/apxs \
--prefix=/opt/cosign  \
--with-cosignconf=/opt/cosign/etc/cosign.conf \
--with-filterdb=/opt/cosign/filter \
--with-cosigndb=/opt/cosign/daemon \
--with-cosigncadir=/opt/cosign/etc/ssl/certs \
--with-cosigncert=/opt/cosign/etc/ssl/certs \
--with-cosignkey=/opt/cosign/etc/ssl/private
```

Přeložíme a zkompilujeme zdrojové soubory:

```
# make everything
```

[1] RC značí Release Candidate - kandidát na konečnou verzi programu
[2] Apache Portable Runtime

Provedeme instalaci nově vzniklých částí:

```
        # make install-all
```

V tuto chvíli jsou serverová část Cosign infrastruktury i filtr v podobě modulu do serveru Apache připraveny k provozu, je třeba je ještě správně nakonfigurovat.

5.1.2 Konfigurace serverové části Cosign

Pro komunikaci pomocí protokolu SSL/TLS bude každá část cosign infrastruktury potřebovat certifikát podepsaný platnou certifikační autoritou[3]. Pro Cosign systém si vytvoříme vlastní certifikační autoritu:

```
# příprava OpenSSL
mkdir -p -m 755 /opt/cosign/certs
cd /opt/cosign/certs
umask 0027          # zákaz čtení cosign klíčů
mkdir -m 700 demoCA
pushd demoCA
mkdir -m 755 newcerts
mkdir -m 700 private
echo "01" > serial
touch index.txt
popd

# vygenerujeme bezpečné heslo
pwgen -s1 > pass.txt

# Kořenová CA
openssl req -new -subj "/C=CZ/SP=Morava/L=Brno/O=VUT/OU=FIT \
        /CN=Root CA/" \
        -x509 -days 3650 -keyout demoCA/private/cakey.pem \
        -out demoCA/cacert.pem -passout file:pass.txt

chmod a+r demoCA/cacert.pem
```

Pomocí OpenSSL vygenerujeme certifikáty pro Cosign systém:

```
# certifikát pro cosignd
openssl req -new -subj "/C=CZ/SP=Morava/L=Brno/O=VUT/OU=FIT \
        /CN=cosignd.local/" -nodes -keyout "cosignd.key" \
        -out "cosignd.csr"
# podepíšeme certifikát
openssl ca -in "cosignd.csr" -out "cosignd.crt" -days 3650 \
        -batch -passin file:pass.txt

# certifikát pro CGI
openssl req -new -subj "/C=CZ/SP=Morava/L=Brno/O=VUT/OU=FIT \
        /CN=cgi-1/" -nodes -keyout "cgi.key" -out "cgi.csr"
# podepíšeme certifikát
openssl ca -in "cgi.csr" -out "cgi.crt" -days 3650 -batch \
        -passin file:pass.txt
chgrp www-data cgi.key cgi.crt
```

[3] v našem případě bude postačovat self-signed certifikát

```
# certifikát pro mod_cosign
openssl req -new -subj "/C=CZ/SP=Morava/L=Brno/O=VUT/OU=FIT \
        /=mod_cosign/" \
          -nodes -keyout "mod_cosign.key" -out "mod_cosign.csr"
# podepíšeme certifikát
openssl ca -in "mod_cosign.csr" -out "mod_cosign.crt" \
        -days 3650 -batch -passin file:pass.txt

# certifikát pro php_filter
openssl req -new -subj "/C=CZ/SP=Morava/L=Brno/O=VUT/OU=FIT \
        /=php_filter/" -nodes -keyout "php_filter.key" \
        -out "php_filter.csr"
# podepíšeme certifikát
openssl ca -in "php_filter.csr" -out "php_filter.crt" \
        -days 3650 -batch -passin file:pass.txt
```

Je třeba vytvořit konfigurační soubor `/opt/cosign/etc/cosign.conf` (význam příkazů je popsán v [17]) a nastavit potřebné direktivy:

```
# certifikát pro komunikaci démona s ostatními účastníky
set        cosigncert         /opt/cosign/certs/cosignd.crt
set        cosignkey          /opt/cosign/certs/cosignd.key
set        cosigncadir        /opt/cosign/certs/CA
# databáze
set        cosigndb           /opt/cosign/daemon
set        cosignhost         cosignd.local
# adresy služeb
set        cosignlogouturl    https://weblogin.local/
set        cosignloopurl      https://weblogin.local/cosign/looping.html
set        cosignnettimeout   300

# povolení přístupu službám s danými údaji v CN cerfitikátu
cgi     cgi-1
service mod_cosign         PT    /opt/cosign/etc/proxy.conf
# připravíme i přístup pro PHP filtr
service php_filter         PT    /opt/cosign/etc/proxy.conf

# autentizační služby
factor /opt/cosign/authn/auth.sh        login password
factor /opt/cosign/authn/pass.sh        passcode
```

Cosign nabízí možnost externí autentizace. Nastavení autentizace v cosign.conf se provádí pomocí direktivy factor. První parametr uvádí cestu k externímu programu, který provádí autentizaci. Další parametry jsou proměnné odeslané z přihlašovacího formuláře (který lze upravit podle potřeby). Tyto proměnné se předávájí na standardní vstup externího programu, po jedné na řádce a v pořadí, v jakém jsou uvedeny v konfiguračním souboru.

V případě úspěšné autentizace musí externí program vypsat název faktoru a skončit s návratovou hodnotou 0. V opačném případě musí skončit s hodnotou 1 a může na standardní výstup zobrazit chybové hlášení.

Příklad externího skriptu pro autentizaci:

```
#!/bin/bash
# Read login and password
```

```
read login
read password
# Check login and password
if [ $login = 'admin' -a $password = 'pass' ]; then
  # Return factor name:
  echo "plain-junk"
  exit 0
else
  # The login form will print this error message:
  echo "$login typed $password.  Try again!"
  exit 1
fi
```

5.1.3 Konfigurace Apache filtru

Do konfiguračního souboru `httpd.conf` serveru Apache byla v průběhu instalace přidána direktiva k použití modulu `mod_cosign`:

```
...
LoadModule cosign_module      modules/mod_cosign.so
...
```

Do stejného konfiguračního souboru poté můžeme zapisovat konfigurační direktivy pro Cosign modul. Například pro vyčleněný virtuální server groupware.local:

```
...
<VirtualHost *:80>
  ServerName groupware.local

  CosignFilterDB          /opt/cosign/filter
  CosignHostname          cosignd.local
  CosignRedirect          https://weblogin.local/
  CosignPostErrorRedirect https://weblogin.local/cosign/post_error.html
  CosignCookieExpireTime  3600
  CosignCrypto            /opt/cosign/certs/mod_cosign.key
                          /opt/cosign/certs/mod_cosign.crt
                          /opt/cosign/certs/CA
  # Don't redirect to https if we come from http
  CosignHttpOnly On

  #proxy tests
  CosignProxyDB           /opt/cosign/proxyDB
  CosignGetProxyCookies On

  #kerberos directives
  CosignTicketPrefix            /opt/cosign/krb_proxy/
  CosignGetKerberosTickets      On

  # Example 1: the root directory is protected
  DocumentRoot /usr/local/apache2/groupware/
  <Directory "/usr/local/apache2/groupware/">
    CosignProtected On
    # Cookie name:
    CosignService groupware
  </Directory>
```

```
64 </VirtualHost>
65 ...
```

Přidáním direktivy do souboru `/etc/services` zařadíme port pro Cosign na seznam známých služeb:

```
echo "cosign 6663/tcp" >> /etc/services
```

Vytvoříme adresáře pro ukládání cookies serveru a filtru. Vytvoříme i adresáře pro ukládání kerberos ticketů. Nastavíme vlastníka a přístupová práva pro tyto adresáře:

```
mkdir daemon filter proxyDB krb_proxy
chown www-data:www-data filter proxyDB krb_proxy
chmod 700 daemon filter proxyDB krb_proxy
```

Nastartujeme Apache server a Cosign démony cosignd a monster:

```
/usr/local/apache2/bin/apachectl -k start
/opt/cosign/sbin/cosignd
/opt/cosign/sbin/monster
```

Při pokusu o připojení ke `groupware.local` jsme přesměrování na přihlašovací adresu `weblogin.local`. Po zadání správných přihlašovacích údajů jsme přesměrováni zpět na doménu `groupware.local`, kde je nám již zpřístupněn chráněný obsah.

5.2 Návrh implementace PHP filtru

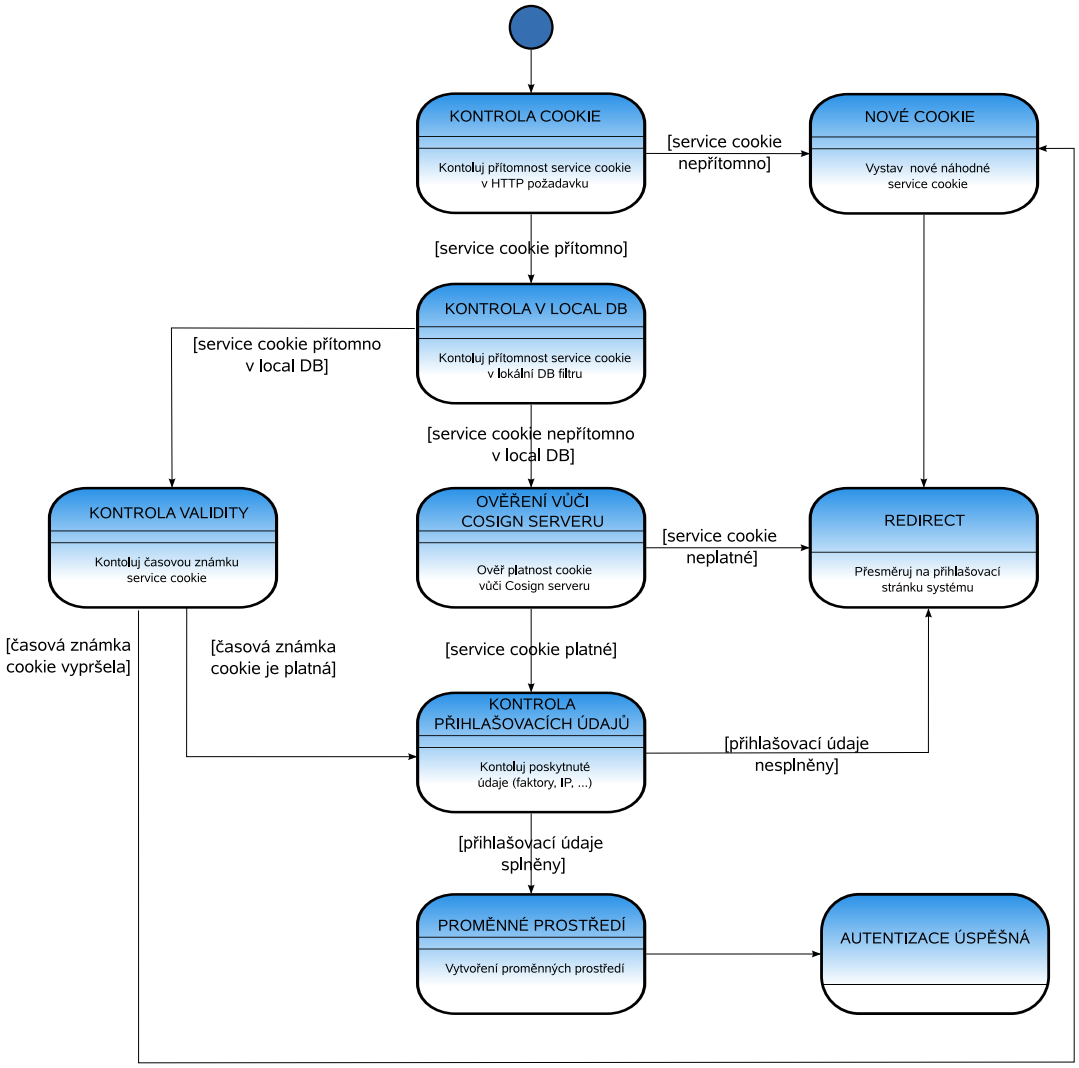
PHP implementace filtru Cosign by měla plně odpovídat specifikaci vydané UMICH[2], odpovídající implementaci filtru pro Apache `mod_cosign`. Dokument obsahuje neformální popis chování filtru, rozdělení činnosti filtru do vrstev a popis konfiguračních direktiv, podle kterých se chování filtru řídí. Filtr prochází postupně několika stavy, jak je zobrazeno na obrázku 5.1.

5.3 Popis implementovaného filtru

K implementaci bylo použito objektově orientovaného přístupu. Užití objektově orientovaného programování (OOP) je vhodné pro intuitivní a přehledný kód, umožňuje navíc snadnou škálovatelnost programu.

Hlavním zdrojem informací pro vývoj filtru tvořila specifikace filtru pro server Apache[2]. Je třeba zmínit, že dokument není právě nejaktuálnější a chování filtru je popsáno neformálně, důležitý zdroj tedy představoval samotný zdrojový text modulu psaný v jazyce C[12]. V těchto zdrojových textech jsem tedy hledal pomoc v případě, že specifikace filtru nenabízela podrobnější informace.

Aby byla ulehčena práce administrátorů pracujících s navrhovaným filtrem, bude ve filtru použit systém oznamování chyb a výjimek s jejich následnou možností zápisu do logovacího souboru. Ve specifikaci není taktéž nic řečeno o verzi komunikačního protokolu, která musí být implementována. Filtr bude implementovat obě verze komunikačního protokolu, jejich výběr bude možno nastavit přepínačem v konfiguračním souboru.



Obrázek 5.1: Stavový diagram implementovaného filtru.

Požadované chování filtru lze nastavit pomocí změny direktiv v konfiguračních souborech. V instalačním adresáři filtru lze specifikovat globální konfigurační soubory, obsahující základní požadovaná nastavení pro použité PHP filtry. Každý PHP skript využívající knihovnu filtru pak může být volitelně přenastaven pomocí lokálního konfiguračního souboru. V lokálním konfiguračním souboru lze nastavit, který globální konfigurační soubor se má použít pro získání informací o požadovaném chování filtru. Pokud by toto zprostředkované nastavení neumožňovalo dostatečnou flexibilitu, je možné v lokálním konfiguračním filtru všechny globální direktivy potlačit a nahradit lokálními.

Filtr je implementován jako knihovna, k jeho použití je tedy třeba přiložit hlavní třídu filtru k aktuálně vytvářenemu projektu[4]. Hlavní třída filtru si při vytvoření instance ostatní potřebné části kódu vezme z poskytnuté knihovny. Samostatnou součástí vytvořené knihovny jsou konfigurační soubory.

[4]Příkazem `include` či `require`

5.3.1 Požadavky na běhové prostředí

Implementace využívá kryptografických funkcí rozšíření OpenSSL, které jsou k dispozici od verze PHP 5.1.0. Filtr při svém spuštění kontroluje tyto závislosti a pokud nejsou splněny, zastaví činnost skriptu a vypíše chybové hlášení.

5.3.2 Popis knihovny

Knihovna PHP filtru je tvořen několika třídami. Třída `CosignFilter` je hlavní třídou, jejíž instance postupně volá metody ostatních tříd pro dosažení komunikace se serverem Cosign, k přístupu k lokální databázi, k reprezentaci cookie či výpisu chybových stavů a zápisu logů. Detailní programátorská dokumentace včetně zdrojových kódů je uvedena jako příloha na vloženém CD nosiči.

CosignFilter

Hlavní třída knihovny. Zapouzdřuje funkčnost celého filtru, je-li třeba, vytváří instance pomocných tříd a využívá jejich metod k vykonávání potřebných operací. Obsahuje metody pro kontrolu validity předloženého service cookie vůči lokální databázi a cosign serveru.

CosignCommunicator

Třída starající se o navazování a ukončování komunikace s Cosign serverem. Též nabízí metody pro zasílání dotazů Cosign serveru podle specifikovaného komunikačního protokolu. Implementuje metody pro získávání odpovědi od Cosign serveru.

CosignCookie

Třída reprezentující service cookie generované filtrem a zpracovávané Cosign serverem. Poskytuje metody pro nastavení jména, hodnoty a časového razítka cookie.

CosignDBAccess

Třída poskytující metody pro přístup k databázové vrstvě Cosign filtru[5]. Čtení a zápis service cookies do lokálního filtru databáze, čtení a zápis proxy cookies, zápis Kerberos ticketů.

CosignDebug

Statická třída zajišťující kontrolní výpisy a zápis do log souboru. Třída obstarává výpisy ladících hlášení, výjimek a zápisů do logovacího souboru.

CosignRandom

Třída poskytující statické metody pro generování náhodných řetězců[6]. Metoda pro generování náhodného řetězce vybírá z 64 povolených znaků vždy náhodný znak a řetězí je za sebou dokud požadované délky řetězce. K náhodnému výběru je použito pseudonáhodného generátoru celých čísel, který používá jako seed aktuální čas v sekundách.

[5] DB vrstva je reprezentována pomocí UNIXového souborového systému

[6] Náhrada za OpenSSL generátor náhodných řetězců, jehož podpora v PHP dosud chybí

CosignConfig

Třída zpracovávající informace z konfiguračních souborů a zpřístupňující tyto informace ve svých atributech. Hodnoty atributů třída CosignConfig zpřístupňuje pomocí veřejných statických metod. Třída při načítání hodnot atributů kontroluje jejich syntaktickou, v některých případech i sémantickou[7] správnost. Pokud nastane chyba, generuje třída příslušnou výjimku či chybové hlášení.

CosignException

Třída dědící z obecné třídy `Exception`. Představuje předka konkrétnějších tříd výjimek:

- `CosignInvalidConfigException` - instance třídy je vytvořena pokud dojde k neplatnému nastavení některé z konfiguračních direktiv
- `CosignConnectionException` - instance třídy je vytvořena pokud dojde k chybě při komunikaci s cosign serverem
- `CosignSecurityException` - instance třídy je volána pokud dojde k chybě, která by mohla ohrozit či ohrožuje zabezpečení chráněného obsahu

5.3.3 Konfigurační soubor globalfilter.conf.php

Globální konfigurační soubor příslušící více filtrům, určený k editaci administrátorem. Obsahuje konfigurační direktivy reprezentované asociativním polem, jehož klíčům pojmenovaným podle direktiv filtru je možné přiřazovat různé hodnoty, podle volby nastavení filtru. Konfigurační direktivy PHP filtru se mírně liší od specifikace filtru pro server Apache. Důvodem bylo například odlišné zpracování certifikátů jazykem PHP nebo přidání volitelných konfiguračních voleb pro rozšíření flexibility implementovaného filtru.

5.3.4 Konfigurační soubor localfilter.conf.php

Lokální konfigurační soubor příslušející jednomu určitému filtru. Pravidla pro zápis direktiv mají stejnou syntaxi jako v případě globálního konfiguračního souboru. Pravidla v tomto souboru však mají vyšší prioritu než pravidla v globálním konfiguračním souboru.

5.4 Dokumentace

Protože bude zdrojový kód předán vývojářům infrastruktury Cosign a je pravděpodobné, že bude nadále upravován a vyvíjen, je jeho důležitou částí právě dokumentace. Proto je přímo v kódu popsána jak funkce každé třídy, tak všechny její atributy a metody. Dokumentace kódu odpovídá standardu phpDocumentor formátu, jež je doporučován Zend vývojovým týmem.[8]. K dispozici je též vygenerovaná dokumentace výše zmíněným programem přiložená ve formátu HTML na CD nosiči.

[7] Např. přítomnost souborů a adresářů

[8] Vývojářská společnost vyvíjející mimo jiné překladač jazyka PHP

5.5 Zabezpečení filtru

Databáze filtru je fyzicky uložena v adresářích přístupných http serverem Apache, je tedy nutné odepřít přístup do těchto adresářů nežádoucím osobám. Doporučené zabezpečení spolu s příklady jsou součástí uživatelské dokumentace, která je přístupná jako příloha C a také je uložena na CD nosiči.

Kapitola 6

Testování filtru

6.1 Funkčnost

Implementovaný filtr je třeba otestovat, nejlépe všechna možná nastavení, která podporuje. Proto jsou v globálním konfiguračním souboru vyplněny téměř všechny direktivy tak, aby bylo využito co nejvíce funkcí filtru. V souboru `index.php` nacházejícím se v kořenovém adresáři virtuálního serveru `php1.local` je vytvořena instance třídy `CosingFilter` a volána metoda `EnableFilter()`, která zaručí autentizaci uživatele požadujícího obsah souboru následující za voláním této metody.

Pro konfiguraci filtru na virtuálním serveru `php1.local` je použit i lokální konfigurační soubor, který upravuje některé jeho direktivy.

Chráněný obsah je symbolicky přichystán jako výpis textu a proměnných prostředí, které Cosign filter vytvořil při úspěšné autentizaci uživatele. K dispozici je i jednoduchý formulář, odesílající data metodou POST zpět skriptu `index.php`, kterým bude otestováno přesměrování při vypršení platnosti service cookie.

Zdrojový kód souboru `test_filter.php`:

```
1  <?php
2  require_once './cosign_filter/CosignFilter.class.php';
3
4  $filter = new CosignFilter();
5  $filter->EnableFilter();
6
7  echo "<html><head></head><body><h3>Cosign protected section :)</h3>";
8  echo "Auth type: ".$_SERVER["AUTH_TYPE"]."</br>";
9  echo "Username: ".$_SERVER["REMOTE_USER"]."</br>";
10 echo "Cosign service: ".$_SERVER["COSIGN_SERVICE"]."</br>";
11 echo "Cosign factor: ".$_SERVER["COSIGN_FACTOR"]."</br>";
12 echo "Remote realm: ".$_SERVER["REMOTE_REALM"]."</br>";
13
14 $sent_data="";
15 if(isset($_POST['data'])){
16         $sent_data = $_POST['data'];
17         echo "<h4>Data sent:$sent_data</h4>";
18 }
19 echo "<form method=\"post\" action=\"test_filter.php\">";
20 echo "<input type=\"text\" name=\"data\" value=\"$sent_data\"><input type=\"submit\"></form> ";
21 echo "</body><html>";
22 $?>
```

6.1.1 Konfigurace

Filtr na serveru `php1.local` je pro testování nastaven tak, aby požadoval tyto parametry:

- Ke komunikaci s cosign serverum bude použit protokol Cosign verze 1
- Cookies budou ověřována oproti Cosign serveru umístěnému na adrese'cosignd.local'!
- Pro jméno service cookie bude použit řetězec `'php_filter1'`
- Hostname pro přihlášení je `'https://weblogin.local'`
- Použit adresář `'/opt/cosign_filter/filterDB/'` k uložení service cookies
- Použit certifikát `'/opt/cosign_filter/certs/php_filter.pem'` pro navázání SSL komunikace
- Platnost cookie vyprší po časovém intervalu 60 sekund
- Filtrem provedené operace jsou zaznamenány do souboru `'/opt/cosign_filter/log/filter1.log'`
- Filtrem provedené operace jsou vypsány na obrazovku
- Pokud je zaslán POST požadavek po vypršení platnosti cookie, je prohlížeč přesměrován na adresu = `'https://weblogin.local/cosign/post_error.html'`
- Po prvotním přihlášení ke službě bude uživatel přesměrován na URL `'http://www.fit.vutbr.cz'`
- Při každém požadavku bude provedena kontrola IP adresy
- Service cookie bude do databáze filtru ukládáno do podadresáře, jehož hash bude mít délku 2 znaky
- Od Cosign serveru budou požadovány proxy cookies
- Proxy cookies budou ukládány do adresáře `'/opt/cosign_filter/proxyDB/'`

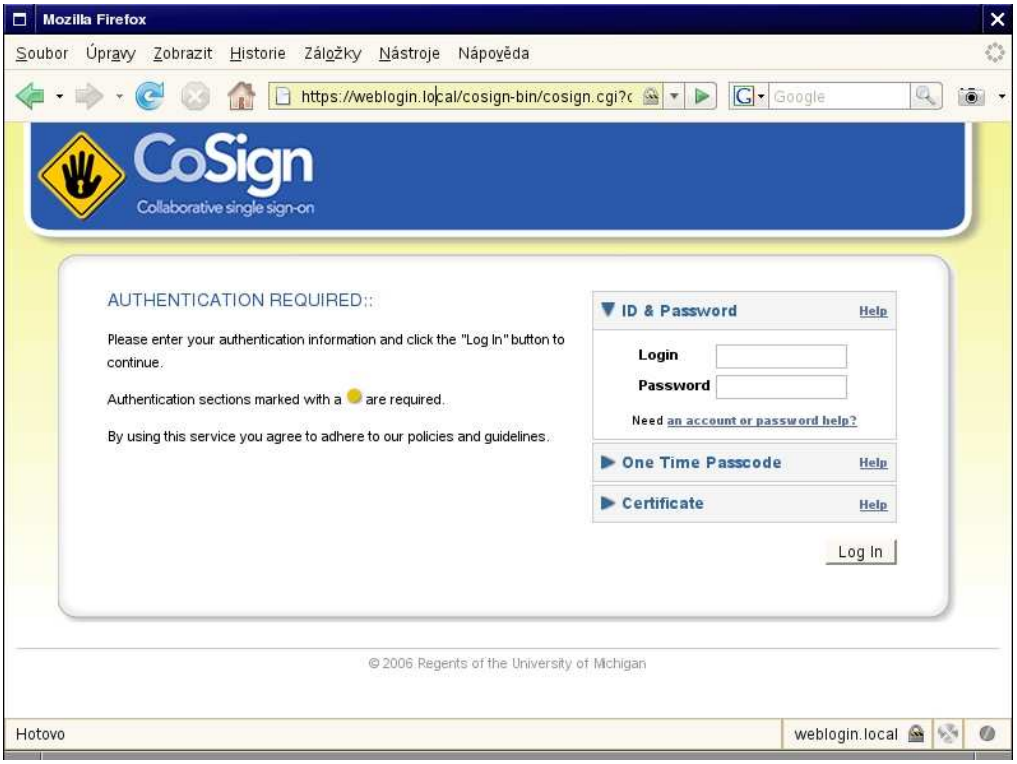
Dále bude nastaven filtr pro soubor `index.php` na virtuálním serveru `php2.local` stejně jako předchozí filtr, jen s těmito rozdíly:

- Ke komunikaci s cosign serverum bude použit protokol Cosign verze 2
- Pro jméno service cookie bude použit řetězec `'php_filter2'`
- Filtrem provedené operace jsou zaznamenány do souboru `'/opt/cosign_filter/log/filter2.log'`
- Po prvotním přihlášení nedojde k přesměrování
- Od Cosign serveru nebudou požadovány proxy cookies
- Filtr bude vyžadovat splnění faktorů `plain` a `otp`
- Filtr bude ignorovat případný suffix `-junk` u splněných faktorů

6.1.2 Průběh činnosti filtru

1. Přístup na adresu `http://php1.local/` - Proběhne přesměrování na adresu `https://weblogin.local`, viz obrázek 6.1. Autentizační služba `weblogin.local` požaduje zadání přihlašovacích údajů. Výpis `filter1.log` souboru:

```
May 11 14:17:35 CosignFilter: enabling filter.
May 11 14:17:35 CosignFilter: Cookie not present in HTTP request -> SETTING new
                              cookie and REDIRECTING to weblogin URL.
```



Obrázek 6.1: Při přístupu na chráněný zdroj je prohlížeč přesměrován na adresu autentizačního rozhraní.

2. Zadáme přihlašovací údaje, pokud jsou správné, autentizační server přesměruje prohlížeč na zpětnou adresu, v našem případě `http://www.fit.vutbr.cz`. Přesměrování proběhlo v pořádku.

3. Vrátíme se na adresu chráněného zdroje `http://php1.local/`. Prohlížeč poskytne filtru service cookie. Filtr se pokusí toto cookie vyhledat ve své lokální databázi. Pokud neuspěje, což je náš případ, spojí se s cosign serverem a požádá o ověření tohoto cookie. Cookie je platné, cosign server zasílá informace o azutentizovaném uživateli. Autentizační služba poskytla faktor `plain`. Souhlasí i IP adresa browseru, kterým přistupujeme k chráněnému zdroji. Výpis části log souboru:

```
...
May 11 14:18:17 CosignFilter: enabling filter.
```

```
May 11 14:18:17 CosingFilter: Service cookie present in HTTP request
May 11 14:18:17 CosignFilter: REQUEST_METHOD - GET
May 11 14:18:17 CosignCommunicator: Connecting to cosignd server.
May 11 14:18:17 CosignCommunicator: Server response: 220 2 Collaborative Web Single Sign-On
May 11 14:18:17 CosignCommunicator: Sending STARTTLS request.
May 11 14:18:17 CosignCommunicator: Server response: 220 Ready to start TLS
May 11 14:18:17 CosignCommunicator: Sending CHECK request.
May 11 14:18:17 CosignCommunicator: Disconnecting from cosignd server.
May 11 14:18:17 CosignFilter: Processing response from cosignd server - cookie valid.
May 11 14:18:17 CosignFilter: Required factors:
May 11 14:18:17 CosignFilter: Check IP passed.
May 11 14:18:17 CosignCommunicator: Connecting to cosignd server.
May 11 14:18:17 CosignCommunicator: Server response: 220 2 Collaborative Web Single Sign-On
May 11 14:18:17 CosignCommunicator: Sending STARTTLS request.
May 11 14:18:17 CosignCommunicator: Server response: 220 Ready to start TLS
May 11 14:18:17 CosignCommunicator: Sending RETRIEVE proxy cookies request.
May 11 14:18:17 CosignCommunicator: Disconnecting from cosignd server.
May 11 14:18:17 CosignFilter: writing proxy cookies into filter proxyDB
May 11 14:18:17 CosignFilter: Write cookie in filter DB.
May 11 14:18:17 CosignFilter: Creating environment variables.

```

4. Prozkoumáním určených adresářů lze zjistit, že service-cookie bylo uloženo do souboru `/opt/cosign_filter/filterDB/wO/cosign-php_filter=wOgUad...DOVNZt`[1], kde podadresář `wO` je dva Byty dlouhý hash ze začátku jména service cookie. Obsah záznamu:

```
i127.0.0.1
padmin
rplain
fplain
```

5. Do adresáře určeného pro ukládání proxy cookies byl uložen záznam `cosign-php_filter=wOgUad...DOVNZt` s tímto obsahem:

```
xcosign-list=OVoeIK...qjQEhR portal.local
xcosign-module=wLlABl...mVkbtl module.local
```

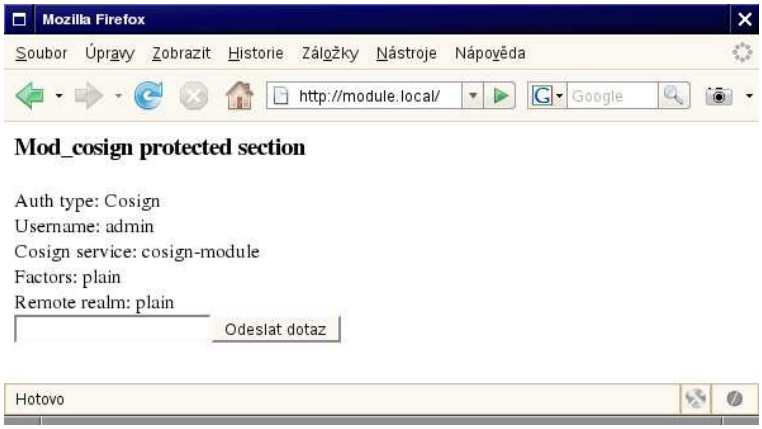
6. Pokusíme-li se znovu přistoupit ke stejné službě, filtr nejdříve ověří, zda je předložené service cookie v jeho cache databázi. Pokud je cookie v cache databázi filtru platné, není nutné ověřovat jeho platnost u cosign serveru a chráněná data jsou zpřístupněna. V našem případě by bylo cookie zneplatněno po 60 vteřinách (nastaveno v konfiguračním souboru). Do logovacího souboru `filter1.log` byly připojeny tyto informace:

```
...
May 11 14:18:35 CosignFilter: enabling filter.
May 11 14:18:35 CosingFilter: Service cookie present in HTTP request
May 11 14:18:35 CosignFilter: Cookie ./cosign_filter/filterDB/wO/
                cosign-php_filter=wOgUadSQMBiDhJqSdfDOVNZtJDDi4eaiKPnyRhx3V
                8BGHOxuhYbfMzplNOGQTJDbleWQzk54rTTeMsScphm3M4JoVk5rgfE4LQbc
                rwH1QGmR8PWTRlnXQqpUMh-sbsrt
```

[1] Jména souborů obsahujících tělo cookie jsou v textu zkrácena

```
38                  exists in filter DB.
39  May 11 14:18:35 CosignFilter: Cookie timestamp valid.
40  May 11 14:18:35 CosignFilter: Reading authN data from filter DB.
41  May 11 14:18:35 CosignFilter: Check IP passed.
42  May 11 14:18:35 CosignFilter: Creating environment variables.
```

7. Abychom ověřili funkčnost single sign-on, zkusíme přistoupit k virtuálnímu serveru `module.local`, který je také chráněn Cosign infrastrukturou, avšak filtrem v podobě modulu `mod_cosign`. Prohlížeč však doméně `module.local` neposkytne vydané service cookie, filtr tedy přesměruje lokaci na adresu přihlašovacího serveru `weblogin.local`. Tomuto je poskytnuto login cookie, které bylo prohlížeči přiděleno při autentizaci pro `php1.local` doménu. Přihlašovací CGI skript rozezná platné login cookie, asociuje jej s vydaným service cookie pro `module.local` a přesměruje zpět na adresu služby `module.local`. Filtr chránící virtuální server `module.local` opět zjistí přítomnost service-cookie, najde jej a ověří jeho platnost vůči cosign serveru. Jelikož bylo toto service cookie v minulém kroku registrováno s platným login cookie, povolí filtr přístup ke chráněným datům, viz obrázek 6.2.



Obrázek 6.2: Při přístupu k jinému zdroji chráněnému toutéž infrastrukturou Cosign je povolen přístup na základě předchozí úspěšné autentizace.

8. Abychom otestovali i možnosti verze 2 Cosign protokolu, zkusíme přistoupit i ke třetímu zdroji `php2.local`, tentokrát opět chráněnému pomocí PHP filtru.

9. Filtr je nastaven na komunikaci pomocí protokolu verze 2, což znamená, že může požadovat autentizaci pomocí více faktorů. Požadované faktory však při přístupu na chráněný zdroj nejsou splněny a prohlížeč je přesměrován na adresu `weblogin.local`. Řádky 18-21 souboru filter2.log to potvrzují:

```
1  May 11 14:21:05 CosignFilter: enabling filter.
2  May 11 14:21:05 CosingFilter: Service cookie present in HTTP request
3  May 11 14:21:05 CosignFilter: Cookie /opt/cosign_filter/filterDB/h3/cosign-
4                  php_filter2=h35YCmXP8ZspK6pJu8PsEgV2KCx35a9wAfc-y-6pE1CvFsy
5                  Sh1Uz3rtM1H8a8yR-XhiXfIdxmLU3LQgY+5jU4iD9ZDRgKdLwFOOmMluyx-
6                  -OwcqjaqqjweDnvgAEiZeM
7                  exists in filter DB.
```

```
May 11 14:21:05 CosignFilter: Cookie timestamp expired.
May 11 14:21:05 CosignFilter: REQUEST_METHOD - GET
May 11 14:21:05 CosignCommunicator: Connecting to cosignd server.
May 11 14:21:05 CosignCommunicator: Server response: 220 2 Collaborative Web Single Sign-On
May 11 14:21:05 CosignCommunicator: Sending STARTTLS 2 request.
May 11 14:21:05 CosignCommunicator: Server response: 220 Ready to start TLS
May 11 14:21:06 CosignCommunicator: Server response: 221 TLS successfully started.
May 11 14:21:06 CosignCommunicator: Sending CHECK request.
May 11 14:21:06 CosignCommunicator: Disconnecting from cosignd server.
May 11 14:21:06 CosignFilter: Processing response from cosignd server - cookie valid.
May 11 14:21:06 CosignFilter: Required factors: plain otp
May 11 14:21:06 CosignFilter: CosigndReturnFactors: plain
May 11 14:21:06 CosignFilter: CosignReturnFactors after SuffixIgnore: plain
May 11 14:21:06 CosignFilter: Required value 'otp' not in returned factors
May 11 14:21:06 CosignFilter: Factors not satisfied -> SETTING cookie and REDIRECTING.
May 11 14:21:06 CosignFilter: Redirect, using HttpOnly
```

10. Filtr zaslal přihlašovací stránce seznam požadovaných faktorů, takže ve formuláři jsou zvýrazněna vstupní pole, které je nutné vyplnit. Pokud tedy správně zadáme přihlašovací jméno, heslo a one time password, bude nám povolen přístup na virtuál `php2.local`.

11. Po úspěšném přihlášení jsme přesměrováni zpět na `php2.local`, kontrolní výpis filter2.log souboru potvrzuje ověření service cookie:

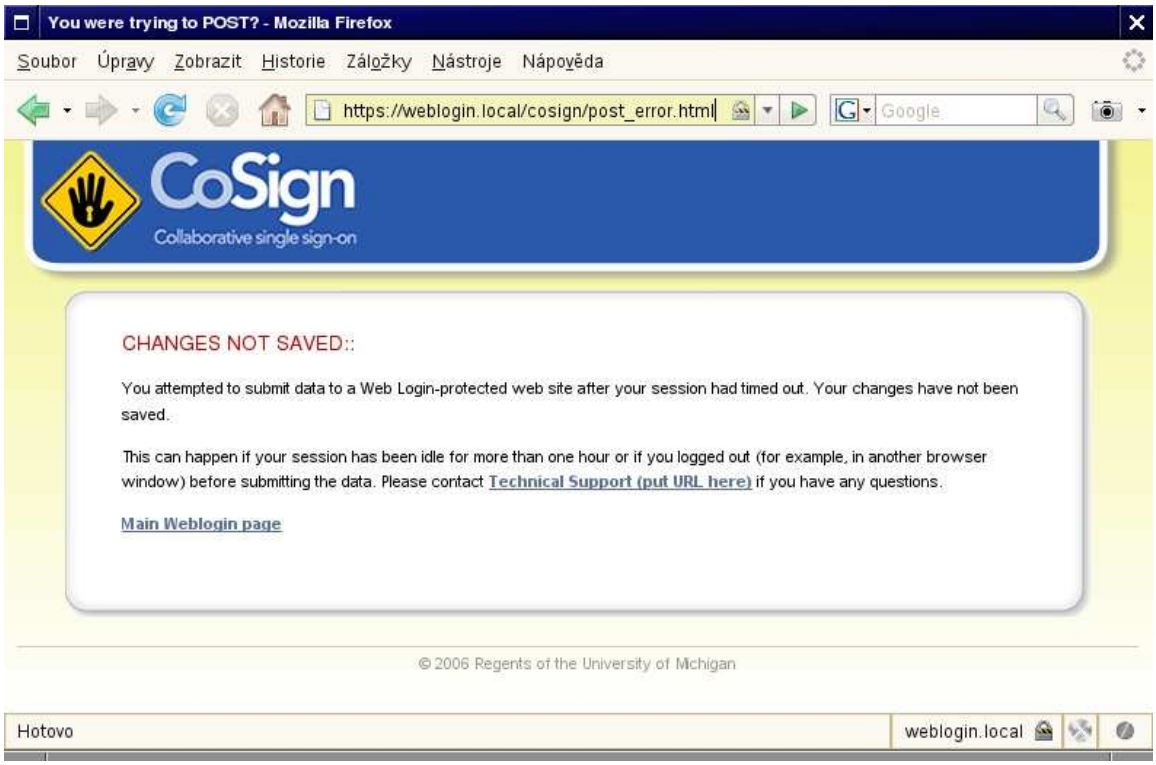
```
May 11 14:25:11 CosignFilter: enabling filter.
May 11 14:25:11 CosingFilter: Service cookie present in HTTP request
May 11 14:25:11 CosignFilter: REQUEST_METHOD - GET
May 11 14:25:11 CosignCommunicator: Connecting to cosignd server.
May 11 14:25:11 CosignCommunicator: Server response: 220 2 Collaborative Web Single Sign-On
May 11 14:25:11 CosignCommunicator: Sending STARTTLS 2 request.
May 11 14:25:11 CosignCommunicator: Server response: 220 Ready to start TLS
May 11 14:25:11 CosignCommunicator: Server response: 221 TLS successfully started.
May 11 14:25:11 CosignCommunicator: Sending CHECK request.
May 11 14:25:11 CosignCommunicator: Disconnecting from cosignd server.
May 11 14:25:11 CosignFilter: Processing response from cosignd server - cookie valid.
May 11 14:25:11 CosignFilter: Required factors: plain otp
May 11 14:25:11 CosignFilter: CosigndReturnFactors: plain otp-junk
May 11 14:25:11 CosignFilter: Ignore factor suffix: -junk
May 11 14:25:11 CosignFilter: CosignReturnFactors after SuffixIgnore: plain otp
May 11 14:25:11 CosignFilter: Check IP passed.
May 11 14:25:11 CosignCommunicator: Connecting to cosignd server.
May 11 14:25:11 CosignCommunicator: Server response: 220 2 Collaborative Web Single Sign-On
May 11 14:25:11 CosignCommunicator: Sending STARTTLS 2 request.
May 11 14:25:11 CosignCommunicator: Server response: 220 Ready to start TLS
May 11 14:25:11 CosignCommunicator: Server response: 221 TLS successfully started.
May 11 14:25:11 CosignCommunicator: Sending RETRIEVE proxy cookies request.
May 11 14:25:11 CosignCommunicator: Disconnecting from cosignd server.
May 11 14:25:11 CosignFilter: writing proxy cookies into filter proxyDB
May 11 14:25:11 CosignFilter: Write cookie in filter DB.
May 11 14:25:11 CosignFilter: Creating environment variables.
```

12. Pokud se po uplynutí doby 60ti vteřin pokusíme odeslat z chráněného skriptu metodou POST jakákoliv data, filtr detekuje vypršení platnosti service cookie a upozorní nás na možnou ztrátu odeslaných dat, viz obrázek 6.3.

13. Máme-li přiděleno uživatelské jméno a heslo pro přístup k systému Kerberos a povolíme-li příslušnou direktivu v konfiguračním souboru, lze od Cosign serveru získat a ukládat Kerberos proxy tickety:

```
...
May 12 22:07:13 CosignCommunicator: Connecting to cosignd server.
May 12 22:07:13 CosignCommunicator: Sending STARTTLS 2 request.
May 12 22:07:13 CosignCommunicator: Sending RETRIEVE kerberos ticket request.
May 12 22:07:13 CosignCommunicator: Reading kerberos ticket, size 459 Bytes
May 12 22:07:13 CosignCommunicator: Disconnecting from cosignd server.
May 12 22:07:13 CosignFilter: writing kerberos ticket into filter krb_proxyDB
May 12 22:07:13 CosignFilter: Write cookie in filter DB.
May 12 22:07:13 CosignFilter: kerberos ticket path:
                ./krbDB/K9oKFzS7Zr74
May 12 22:07:13 CosignFilter: Creating environment variables.
```

14. Cesta ke Kerberos proxy ticketu je poté filtrem zveřejněna pomocí proměnných prostředí a ticket tak lze dále využívat pro autentizacei třetími stranami.



Obrázek 6.3: Přesměrování při odeslání POST požadavku po vypršení platnosti cookie

6.2 Výkonnostní parametry

6.2.1 Porovnání s implementací v jazyce C

K porovnání výkonnosti implementace v PHP a implementace v jazyce C bylo použito programu Apache Benchmark, jenž je součástí instalace Apache serveru. Též bylo použito

testovacího programu Siege, dovolujícího specifikovat vykonávání zátěže serveru po určitou dobu.

Apache Benchmark

Nástroj Apache Benchmark je součástí standardní instalace HTTP serveru Apache, je tudíž vhodný hlavně pro testování výkonnosti Apache serverů. Výsledkem testu pomocí Apache Benchmark je soubor statistik, z nichž nejdůležitější jsou například počet zpracovaných požadavků za vteřinu, průměrná rychlost provedení jednoho požadavku, velikost přenesených dat, kolik připojení se nezdařilo a také jaký podil času připadl na připojování, zpracování požadavku a čekání na odpověď.

Nástroj je vhodný pro zátěžové testování, tzn. že lze nastavit počet cílových požadavků na zdroj a zároveň počet konkurentních požadavků (běžících paralelně). Například pro otestování adresy `http://localhost/` tisíckrát s deseti konkurentními požadavky spustíme příkaz `ab` s těmito parametry:

```
ab -n1000 -c10 http://localhost/
```

V našem případě je třeba zasílat serveru také service cookie pro autentizaci uživatele, pomocí přepínače `-C` lze dodatečná cookie snadno specifikovat.

Siege

Siege je HTTP testovací a zátěžový nástroj navržený pro testování výkonnosti HTTP serverů. Siege podporuje základní autentizaci, cookies a HTTP i HTTPS protokol. Dovoluje uživateli přistupovat k serveru s nastavitelným počtem souběžně simulovaných uživatelů. Testování pomocí Siege bylo zvoleno také pro jeho možnost zátěže serveru po určitý nastavený časový interval a pro možnost simulace náhodného přístupu uživatelů na zvolený seznam zdrojů. Například pro simulaci deseti uživatelů náhodně přistupujících k různým cestám zdroje po dobu pěti minut s prodlevou 0-3 sekundy lze použít tento příkaz:

```
siege -c 10 -i -t 5m -d 3 -f url_list.txt
```

6.2.2 Výsledky testování

Testování proběhlo jak pomocí Apache Benchmarku (počet požadavků byl testován od 1000 do 30000, ovšem tato změna podle předpokladu nijak neovlivnila výkonnost systému, pouze se lineárně prodloužila doba testování), tak i pomocí Siege.

Zátěžové testování na počet požadavků pomocí Apache Benchmark

- 5000 celkových požadavků
- Simulováno 15 souběžných konkurentů

Apache Benchmark - konstantní počet transakcí		
	Mod_cosign	**PHP filter**
Jméno serveru:	module.local	php1.local
Port:	80	80
Cesta dokumentu:	/	/
Délka dokumentu [Byte]:	293	317
Počet konkurentů:	15	15
Délka testu [sekund]:	**5.68**	**50.19**
Celkem požadavků:	5000	5000
Chybných požadavků:	0	0
Write errors:	0	0
Celkem přeneseno: [MB]	2.535	2.655
HTML přeneseno: [MB]	1.465	1.585
Požadavků za sekundu (průměr):	**878.77**	**99.61**
Délka požadavku (průměr) [ms]:	17.069	150.589
Délka požadavku (průměr souběžných):	1.138	10.039
Přenosová rychlost [KBytes/sec]:	434.99	51.64

Zátěžové testování na čas

- Doba testování 3 minuty[2]
- Simulováno 15 souběžných konkurentů

Siege - konstantní doba		
	Mod_cosign	**PHP filter**
Transakcí:	137770	20765
Dostupnost:	99.99%	99.99%
Délka trvání [sekund]:	191.04	208.02
Přenesených dat [MB]:	38.50	6.28
Čas odpovědi (průměr)[sekund]:	0.02	0.14
Transakcí za sekundu (průměr):	**721.16**	**99.82**
Propustnost (průměr)[MB/sekundu]:	0.20	0.03
Souběžných konkurentů:	12.42	14.11
Úspěšných transakcí:	137770	20765
Chybných transakcí:	7	3

Z uvedených výsledků programu Apache Benchmark je zřejmé, že implementace filtru v jazyce C je téměř 9-násobně rychlejší oproti implementaci v PHP. Měření potvrzují i výsledky programu Siege, který dospěl ke stejnému výsledku počtu transakcí/sekundu u PHP filtru. U verze mod_php naměřil Siege mírně odlišné hodnoty, což může být způsobeno tím, že Apache Benchmark je určen přímo pro práci se servery Apache, tudíž je pro toto měření i přesnější, kdežto Siege je určen obecně pro práci s jakýmkoliv typem HTTP serveru[13].

[2]Přestože byla doba u obou filtrů nastavena na 180 vteřin, program ji přesně nedodržel, proto se u obou provedených testování celkové časy mírně liší

Kapitola 7

Závěr

Pojem „bezpečná autentizace“ hraje v dnešní době mnohem větší roli než v počátcích provozu sítě Internet. Množství uživatelů se každým rokem rapidně zvyšuje a s tím narůstají i nároky na systémy prokazující či ověřující jejich identitu. Mechanismus centrální autentizace a s ním spojené jednotné přihlášení mohou významně odlehčit práci jak uživatelům a administrátorům počítačových sítí, tak i zátěži, která je na sítě kladena.

Systém Cosign podle provedeného rozboru nabízí zajímavé prostředky pro realizaci centrální webové autentizace. Také proto byl Fakultou infromačních technologií zvolen jako kandidát, zajišťující přístup ke chráněným zdrojům v síťovém prostředí fakulty. Svou roli při jeho výběru jistě sehrál i fakt, že je šířen jako open-source software a že podporuje aplikaci filtru jak na Linuxových serverch, tak na serverech IIS společnosti Microsoft. Důležitým rysem však je možnost implementovat vlastní filtr podle specifikace zveřejněné týmem vyvíjejícím Cosign na Michiganské univerzitě.

7.1 Dosažené výsledky

Prozkoumáním specifikace vyšlo najevo, že implementace autentizačního filtru v jazyce PHP je možná, vzhledem k rozšířeným kryptografickým možnostem knihoven PHP. Potřebná podpora asymetrické kryptografie je zajištěna knihovnou OpenSSL, symetrické kryptografické funkce nabízí knihovna mcrypt a hashovací funkce jsou v aktuální verzi PHP standardně podporovány bez potřeby přidávání jakýchkoliv knihoven.

Implementovaný filtr plnohodnotně nahrazuje dosud používané filtry pro server Apache či IIS a Java servlety. Instalace a konfigurace PHP filtru je navíc jednodušší, méně náročná na znalosti administrátora a přístupná i pouhým uživatelům webového serveru. PHP filtr též nabízí snadnou volbu pro komunikaci filtru podle Cosign protokolu verze 1 či 2, což u dosavadních filtrů nebylo k dispozici. Filtr byl testován i z hlediska výkonnosti. Oproti implementaci v jazyce C je PHP filtr znatelně pomalejší, což je dáno hlavně faktem, že PHP je skriptovacím jazykem, tudíž jistou dávku času zabere načítání PHP tříd do paměti a samotná interpretace PHP skriptu. Modul mod_cosign je oproti tomu přítomen v paměti od spuštění serveru Apache, navíc je naprogramován v jazyce C, což je samo o sobě jistou zárukou výkonnosti.

7.2 Zhodnocení

Implementace filtru pro systém Cosign ve skriptovacím jazyce PHP má své nesporné výhody, z hlediska výkonnosti je však třeba uvažovat, jaké zátěži může být tento filtr vystaven. Pokud nepůjde o nasazení tohoto filtru v systémech s velmi vysokými požadavky na výkonnost, je možné jej reálně použít.

Ke knihovně filtru je přidána i uživatelská a programátorská dokumentace v anglickém jazyce, zdrojový kód je taktéž pro možnost případného pokračování v projektu vývojovým týmem Cosign dokumentován anglicky. I když neustále probíhá vývoj serverové části Cosign systému, implementace PHP Cosign filtru odpovídá požadovanému protokolu, tudíž komunikuje s různými vývojovými verzemi serveru. Filtr se nachází ve stavu, kdy je možno jej předat vývojovému týmu a začít používat.

Literatura

[1] Central Authentication Service by JASIG. [Online], [cit. 2008-01-07].
URL http://www.ja-sig.org/products/cas/index.html

[2] Cosign filter specification. [Online], Draft XII (April 2006), [cit. 2008-05-14].
URL http://weblogin.org/media/authentication_filter2006a.rtf

[3] Cosign: web single sign-on. [Online], [cit. 2008-05-14].
URL http://www.umich.edu/~umweb/software/cosign/

[4] IT Services: Stanford WebAuth. [Online], [cit. 2008-01-07].
URL http://www.stanford.edu/services/webauth/

[5] Kerberos: The Network Authentication Protocol. [Online], [cit. 2008-05-14].
URL http://web.mit.edu/Kerberos/

[6] PHP: Hypertext Preprocessor. [Online], [cit. 2008-05-17].
URL http://php.net/

[7] Pubcookie Home Page. [Online], [cit. 2008-01-07].
URL http://www.pubcookie.org/

[8] RFC 1704: On Internet Authentication. [Online], [cit. 2008-05-14].
URL http://www.ietf.org/rfc/rfc1704.txt

[9] RFC 2246: The TLS Protocol. [Online], [cit. 2008-01-07].
URL http://www.ietf.org/rfc/rfc2246.txt

[10] RFC 2965: HTTP State Management Mechanism. [Online], [cit. 2008-05-14].
URL http://www.ietf.org/rfc/rfc2965.txt

[11] RFC 4234: Augmented BNF for Syntax Specification: ABNF. [Online], [cit. 2008-05-14].
URL http://www.ietf.org/rfc/rfc4234.txt

[12] Zdrojové kódy Cosign filtru pro server Apache. [Online], Verze 2.0.2a (2007), [cit. 2008-05-14].
URL http://sourceforge.net/projects/cosign

[13] Brittain, J.; Darwin, I. F.: *Tomcat: The Definitive Guide*, kapitola Measuring Web Server Performance. O'Reilly, 2007, ISBN 0-596-10106-6, s. 130–134.

[14] Hanáček, P.; Staudek, J.: *Bezpečnost informačních systémů*. ÚSIS, 2000, ISBN 80-238-5400-3.

[15] Menezes, A. J.; van Oorschot, P. C.; Vanstone, S. A.: *Handbook of Applied Cryptography*. CRC Press, 1996, ISBN 0-8493-8523-7.

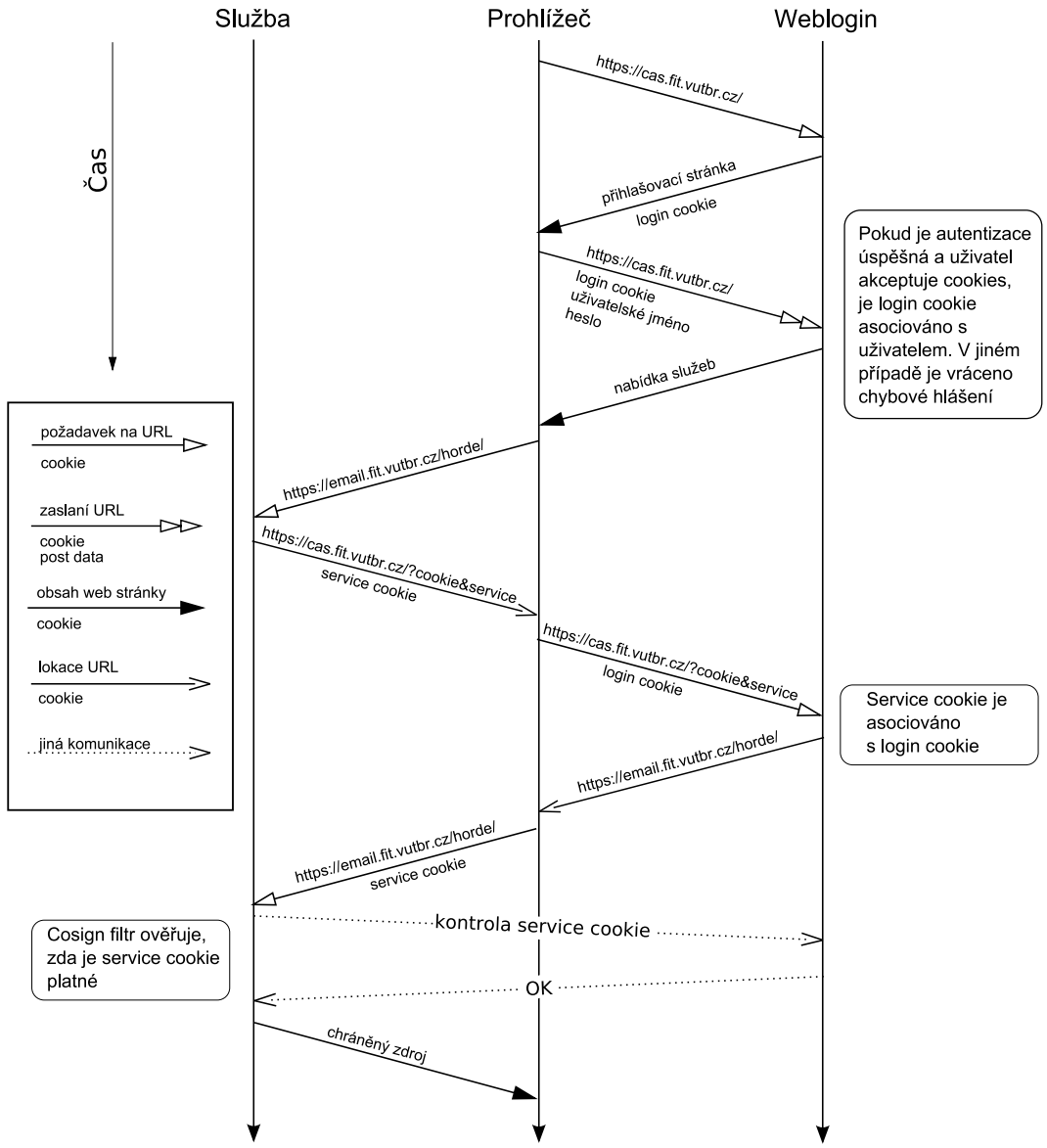
[16] Schlossnagle, G.: *Pokročilé programování v PHP5*. Zoner Press, 2004, ISBN 80-86815-14-5.

[17] University of Michigan: *Cookie Sign-On Daemons and CGI configuration file manual*. 2007.

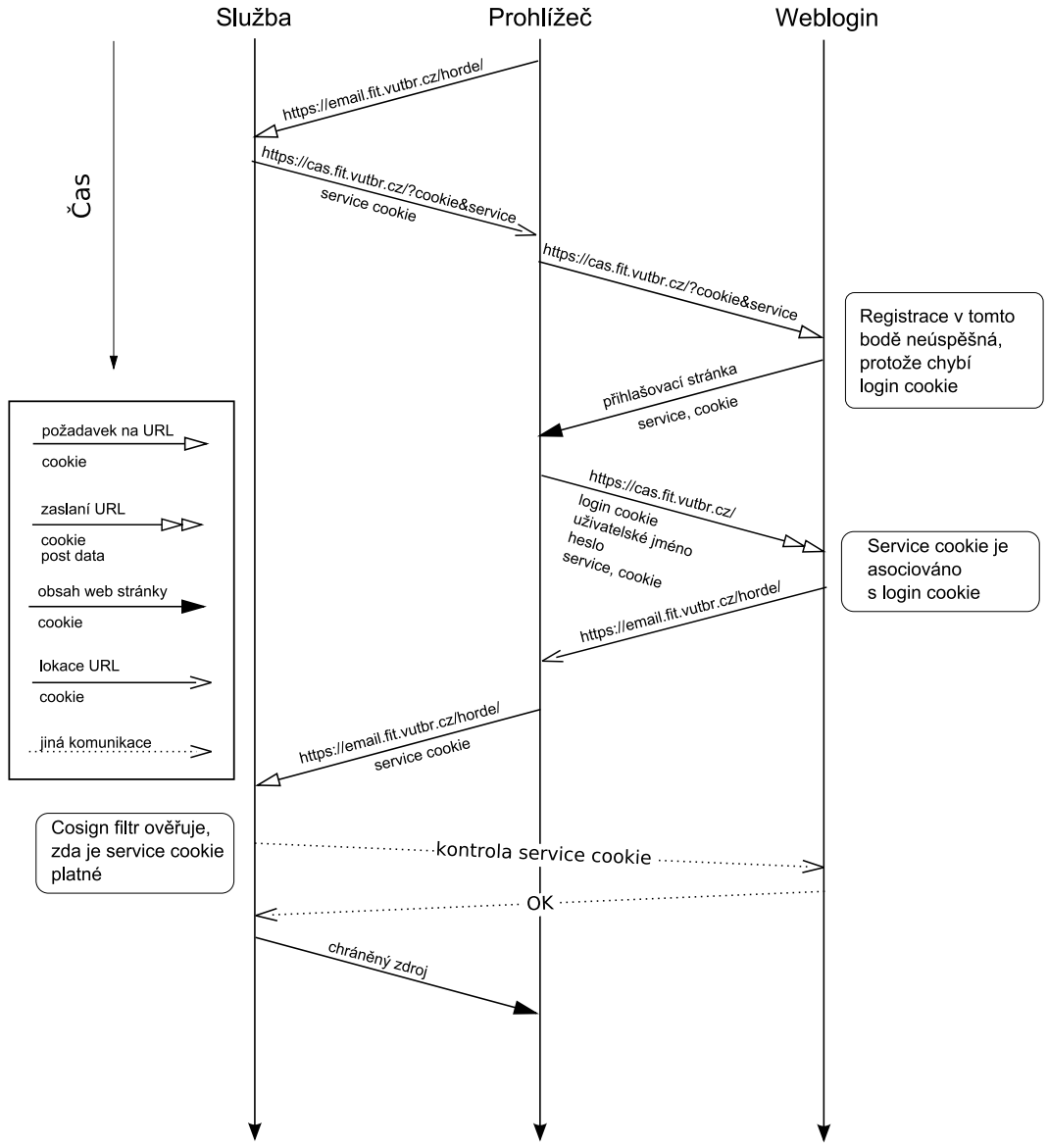
Seznam příloh

1. Příloha A - Příklady komunikačního toku Cosign komunikace
2. Příloha B - Konfigurační direktivy rozeznávané modulem filtru
3. Příloha C - Uživatelská dokumentace pro instalaci a konfiguraci PHP Cosign filtru
4. CD-ROM nosič s následujícími adresáři a obsahem:
    - `/text` - text této práce v elektronických formátech
    - `/cosign_filter/include` - zdrojové kódy knihovny
    - `/cosign_filter/conf` - konfigurační soubory
    - `/cosign_filter/README` - informace o knihovně
    - `/cosign_filter/INSTALL` - uživatelská dokumentace
    - `/doc` - programátorská dokumentace ve formátu HTML

Příloha A



Obrázek 7.1: Příklad komunikačního toku při SSO přihlašování Cosign - uživatel se nejdříve autentizuje na CAS serveru.



Obrázek 7.2: Příklad komunikačního toku při SSO přihlašování Cosign - uživatel se nejdříve pokouší přistoupit k chráněné službě.

Příloha B

Syntaxe: CosignHostname plně-kvalifikované-doménové-jméno
Implicitní hodnota: cosign.example.edu (jméno bude rozhodnuto později)
Popis: jméno hosta, na kterém běží cosignd démon

Syntaxe: CosignPort číslo
Implicitní hodnota: 6663
Popis: port, na kterém Cosign naslouchá požadavky na autentizaci

Syntaxe: CosignService jméno-služby
Implicitní hodnota: nic
Popis: jméno služby, která je zabezpečena filtrem

Syntaxe: CosignRedirect URL
Implicitní hodnota: nic
Popis: URL pro přesměrování na přihlašovací službu, typicky login CGI

Syntaxe: CosignPostErrorRedirect URL
Implicitní hodnota: nic
Popis: URL odkazující na webovou stránku zobrazující se při POST požadavku provedeném po expiraci service cookie

Syntaxe: CosignRequireFactor Faktor1 [Faktor2 FaktorN]
Implicitní hodnota: nic
Popis: seznam faktorů, které musí uživatel splnit

Syntaxe: CosignFactorSuffix FaktorSuffix
Implicitní hodnota: nic
Popis: přípona cosign faktorů

Syntaxe: CosignFactorSuffixIgnore On—Off
Implicitní hodnota: Off
Popis: přepínač indikující ignorování přípon cosign faktorů

Syntaxe: CosignFilterDB Cesta
Implicitní hodnota: /var/cosign/filter
Popis: absolutní cesta k adresáři, v němž bude umístěna lokální databáze service cookies

Syntaxe: CosignProxyDB Cesta
Implicitní hodnota: /var/cosign/proxy
Popis: absolutní cesta k proxy databázi service cookies

Syntaxe: CosignFilterHashLength 0—1—2
Implicitní hodnota: 0
Popis: délka hashe podadresáře pro databázi Cosign filtru

Syntaxe: CosignTicketPrefix Cesta

Implicitní hodnota: /ticket
Popis: cesta ke místu, kde se ukládají Kerberos lístky

Syntaxe: CosignProtected On—Off
Implicitní hodnota: On
Popis: přepínač indikující zabezpečení lokace či adresáře

Syntaxe: CosignSiteEntry URL
Implicitní hodnota: nic
Popis: URL adresa zdroje, na který je uživatel odkázán po přihlášení

Syntaxe: CosignAllowPublicAccess On—Off
Implicitní hodnota: Off
Popis: přepínač nastavující volitelnou autentizaci pro chráněné zdroje

Syntaxe: CosignHttpOnly On—Off
Implicitní hodnota: Off
Popis: přepínač nastavující možnost použití modulu bez SSL

Syntaxe: CosignCrypto <soubor s klíčem> <soubor s certifikátem> <ca adresář>
Implicitní hodnota: /var/cosign/certs/key.pem /var/cosign/certs/cert.pem /var/cosign/certs/CA
Popis: cesty k certifikátům, klíčům a certifikační autoritě

Syntaxe: CosignCookieExpireTime <čas v sekundách>
Implicitní hodnota: 86400 (24 hodin)
Popis: doba platnosti cookie

Syntaxe: CosignGetProxyCookies On—Off
Implicitní hodnota: Off
Popis: nastavuje, zda budou požadovány proxy service cookies

Syntaxe: CosignGetKerberosTickets On—Off
Implicitní hodnota: Off
Popis: přepínač nastavující, zda bude požadován TGT lístek

Syntaxe: CosignKerberosSetupGSS On—Off
Implicitní hodnota: Off
Popis: přepínač upravující nastavení prostředí podle ostatních Apache modulů s ohledem na Kerberos a GSSAPI[1]

Syntaxe: CosignCheckIP never—initial—always
Implicitní hodnota: initial
Popis: hodnota určující, zda se má ověřovat IP adresa klienta vůči IP adrese uložené v databázi

[1]Generic Security Services Application Programming Interface - programové rozhraní pro klient-server autentizaci přes síť

Příloha C

PHP CosignFilter library version 0.1 - INSTALL file

1. Unzip directory with PHP CosignFilter files to desired location, e.g:

```
unzip -d /opt/cosign/ cosignPHPfilter_v0.1.zip
```

2. Edit `globalfilter.conf.php` file in cosign_filter_path directory. Global config can be used by more Cosign PHP filters, setting basic directives. By default, all directives in `globalfilter.conf.php` are available and fully commented. Set at least these directives:

   (a) `CosignHostname` - name of server, where cosignd is running

   (b) `CosignPort` - port on which cosignd server listens

   (c) `CosignService` - name of service cookie for your protected scripts

   (d) `CosgignRedirect` - URL to redirect for login

   (e) `CosignFilterDB` - directory where filter cookies are cached (relative to CosignFilterDirectory)

   (f) `CosignCryptoLocalCert` - path to Cosign filter certificates (relative to CosignFilterDirectory).

   Certificates must be in PEM format, so for example:

```
# Create certificate
openssl req -new -subj "/C=CZ/SP=Moravia/L=Brno/O=FIT/OU=UIFS/CN=php_filter/" \
-nodes -keyout "php_filter.key" -out "php_filter.csr"
# Sign certificate
openssl ca -in "php_filter.csr" -out "php_filter.crt" -days 3650 -batch \
-passin file:your_pass.txt
# Concatenate key and cert to get PEM format
cat php_filter.key >  php_filter.pem
cat php_filter.crt >> php_filter.pem
```

   (g) `CosignCookieExpireTime` - expiration time of service-cookie in seconds

   (h) `CosignProtected` - finally confirm the protection by setting this directive to 'On'

   (i) Optional directives can be set too (recommended are CosignLogUseLog and CosingFilterLogFile) All available directives are commented in `globalconfig.conf.php` file

3. Protect prepared directories that will contain secure data!!! PEM certificate is private information used only by Apache server, so you have to disable access to read it by other users on your machine. If you place some of your private data under Apache document root (e.g. logfile, proxy cookies, ...), don't publish your sites until you make sure, that no one from these elements are accessible by third person. Assume that Apache is running under user wwwroot and group wwwroot (change this according to your Apache user:group):

   (a) Directory with certificates

```
# restrict access
chown wwwroot:wwwroot cert_dir/
chmod 300 cert_dir/

# in case your cert is under document root (not recommended),
  add .htaccess file in cert_dir/ with following content
<Files "*.pem">
  order allow,deny
  deny from all
</Files>
# end htaccess
```

(b) Directory with filterDB

```
# restrict access
chown wwwroot:wwwroot cert_dir/
chmod 300 filterDB/
```

(c) Directory with proxy cookies

```
# restrict access
chown wwwroot:wwwroot cert_dir/
chmod 300 proxyDB/
```

(d) Directory with proxy kerberos tickets

```
# restrict access
chown wwwroot:wwwroot cert_dir/
chmod 300 krbDB/
```

(e) Optionally log directory

```
# restrict access
chown wwwroot:wwwroot cert_dir/
chmod 300 log/
```

4. Create file called 'localfilter.conf.php' in directory containing scripts you want to protect. It is local configuration file used just for this directory. There is only one mandatory directive in this file (directive telling filter where to get basic connection and other settings):

```
1  $cfg['CosignGlobalConfigFile'] = '/path/to/your/global/config/file';
```

You can override other directives from `globalfilter.conf.php` file in this local configuration file - simple by setting them again (filter first processes directives from `globalfilter.conf.php`, then from `localfilter.conf.php`). You can for example use different log and service cookie name for this particular filter:

```
3  $cfg['CosignFilterLogFile']     = '/var/log/my_php_filter1.log';
4  $cfg['CosignService']           = 'php_filter1';
```

5. In your PHP script, you want to protect with Cosign, enable filter:

```
<?php
  require_once '/path/to/your/cosign/filter/directory/include/CosignFilter.class.php';
  $filter = new CosignFilter();   // create new instance of Cosign filter
  $filter->EnableFilter();        // enable Cosign filter for following content
  ?>
  <html>
  <head><title>My protected site</title></head>
  <body>
    <p>My protected site content<p>
  </body>
  </html>
```

6. That's all! Test your script, if any problems appears, any exceptions are written to browser window, output to log file is recommended, so see Cosign filter log file (set its directives in config files).