RYCHLÁ MULTIPLATFORMNÍ AUTENTIZACE V INTERNETU

PETR VOBORNÍK

There are many ways to secure communication between the individual parts of distributed applications. But some of them are not resistant of all types of attacks and some cannot be used multi-platform. Secure authentication is necessary especially when communicating via web services. Some existing solutions will be presented in the article and a new original method will be introduced. The presentation of the results of experimental measurement of the speed of this algorithm will be included.

authentication, hash, multiplatform, password, salt

1. ÚVOD

Při komunikaci distribuovaných aplikací prostřednictvím veřejné sítě internet existuje řada rizik, s nimiž je nezbytné počítat a předcházet jim. Kromě technických úskalí, je zde možnost nežádoucího zásahu třetí osoby – útočníka. Ten může kompletní komunikaci dvou strana odposlouchávat, zaznamenávat, analyzovat, dokonce do ní aktivně vstupovat modifikací zasílaných dat, jejich opakovaným použitím, nebo i podvržením zpráv vlastních.

1.1 Webové služby

Zvláštní pozornost si zaslouží tzv. webové služby. Jde o *softwarové aplikace, ke kterým se vzdáleně přistupuje přes klasické protokoly založené na XML. Webová služba standardně definuje formát zprávy, specifikuje rozraní ve kterém má být zpráva posílána, popisuje konvence pro mapování obsahu zprávy pro vstup a výstup z programů provozujících službu a definuje mechanizmy pro zpřístupňování a zveřejňování rozhraní služby.* [1]

Například webové služby používané v Microsoft .NET Frameworku jsou implementovány jako metody tříd, které mohou být vzdáleně volány libovolnou aplikací, jež dodrží komunikační protokol SOAP[1]. Ten ale standardně neobsahuje žádné zabezpečovací mechanizmy, které by bránily zneužití webové služby nepovolanými osobami či aplikacemi.

Microsoft v knize Web Service Security [2] nabízí několik standardizovaných řešení přímé i zprostředkované autentizace přes WSE[2]. Zprostředkovaná řešení zahrnují Kerberos, X.509 PKI a Security Token Service, které pochopitelně vyžadují pro svůj provoz další externí systémy a oddělenou správu identit uživatelů. V případě přímého řešení je pak správa a ověřování identit součástí nebo podsystémem aplikace přímo za webovou službou. Navrženy jsou implementace tří variant autentizace: pomocí Active Directory (přes LDAP), dle databáze dodržující předdefinované schéma, nebo vytvořením vlastní autentizační třídy. Veškeré identifikátory jsou přitom součástí hlavičky SOAP obálky, takže přímo webová služba není zatěžována dalšími nadbytečnými parametry. Druhé dvě varianty by tedy mohly být univerzálním řešením, avšak automatizace jejich zpracování na klientské straně .NET technologiemi komplikuje využití v klientských aplikacích vytvořených na jiných platformách než .NET.

Přístup článku [1] taktéž využívá hlavičku

[1] SOAP – *Simple Object Access Protocol je protokol založený na XML, který je základem pro multiplatformní a na programovacím jazyku nezávislou komunikaci přes webové služby distribuovaných aplikací* [1]

[2] WSE – Web Services Enhancements – přídavný modul pro Microsoft .NET Framework rozšiřující možnosti webových služeb, mimo jiné o možnost jejich zabezpečení na úrovni SOAP.

SOAP obálky pro zasílání identifikačních údajů o uživateli, ale tato data jsou tam přidávána plně kontrolovaným kódem aplikace. Pro autentizaci je využita ruční implementace SRP[3] protokolu, který si při zahájení spojení se serverem vymění nezbytné údaje, z nichž je vygenerován autentizační „session key“, platný pouze pro aktuální spojení.

Toto řešení není náročné na implementaci, ovšem vyžaduje pouze pro své účely vícero výměn dat mezi serverem a klientem, což zatěžuje server. Článek [1] v závěru uvádí měření, z něhož vyplývá, že použití tohoto postupu zpomaluje zpracování požadavku v průměru 4,74x oproti použití webové služby bez autentizace. Vzhledem k faktu, že tato autentizace není závislá na délce zprávy, je relevantnějším údajem spíše fakt, že zpomalení způsobené touto metodou bylo naměřeno 1 - 1,5 s, přičemž vliv na tuto dobu má i rychlost spojení se serverem, který byl však při tomto pokusu na témže počítači jako klientská aplikace.

Článek [3] navrhuje robustní bezpečnou autentizaci na základě hashe hesla zkombinovaného s náhodnou hodnotou (salt). Heslo ani jeho hash přitom nemusí nikdy putovat přes veřejnou síť, dokonce ani při prvotní registraci uživatele, podmínkou však je uchování nejen hesla, ale i složitých hash-kódů, buď v certifikačním souboru, nebo na čipové kartě. Tato podmínka velmi komplikuje využití pro aplikace založené na cloud computingu, jež by měly být použitelné kdykoli a odkudkoli, neboť je zapotřebí mít u sebe neustále soubor s certifikátem a nestačí pouze znát zvolené heslo.

Uvedená řešení tedy požadavku univerzálnosti nevyhovují zcela, navíc u nich nelze vyloučit nežádoucí posílání starších zpráv na server útočníkem typu man-in-the-middle.

2. UNIVERZÁLNÍ AUTENTIZACE

Nový, multiplatformní autentizační postup byl již představen v článku [4]. Jeho podstatou je tzv. ticket, což je bez znalosti hesla nezfalšovatelný kontrolní součet („podpis“) unikátní pro každou komunikaci a to i v případě, že zasílaná data budou stále stejná. Aby bylo při autentizaci vždy dosaženo jeho jedinečnosti je pro každou komunikaci znovu vygenerován nový unikátní salt[4], např. pomocí tzv. GUID hodnoty, u níž je pravděpodobnost opakování téměř nemožná [5]. Výpočet kontrolního ticketu pro aktuální komunikaci se serverem proběhne jako hash z kombinace hashe hesla uživatele a unikátního saltu.

Salt a vypočtený kontrolní ticket jsou poté přiloženy spolu s daty ke zprávě a ta je odeslána na server. Celý postup ukazuje Obrázek 1. Jelikož útočník hash hesla, jež není součástí komunikace, nezná, ani jej nemůže zachytit nebo zpětně vypočítat, nedokáže kontrolní ticket vytvořit.

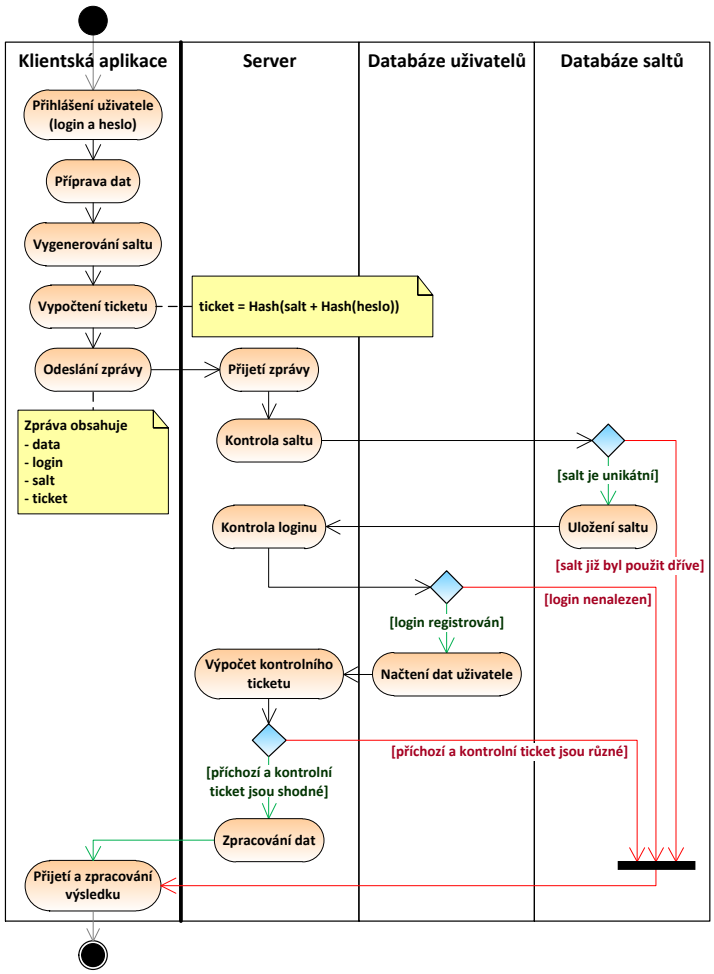
Pro ověření unikátnosti saltu stačí na serveru např. jedna databázová tabulka o jednom sloupci, který je zároveň jejím primárním klíčem, kam se ukládají veškeré příchozí salty. Možné je také tabulku rozšířit o další sloupce a evidovat v ní i jiné „logovací“ údaje (např. ID uživatele, IP, časovou značku, …). Některé z těchto informací mohou být i obsaženy přímo v saltu. Třeba časová značka skombinovaná s ID uživatele by se s velkou pravděpodobností již neměla nikdy opakovat.

2.1 Rychlost

Časová náročnost tohoto způsobu autentizace byla ověřena měřením na testovací aplikaci vytvořené na platformě Silverlight. Ta se připojovala k serveru, vytvořenému v ASP.NET, pomocí klasické webové služby (asmx). Databázový systém na serveru byl v tomto případě použit Firebird 2.5. Pro výpočet hashe a generování GUID byly na obou stranách použity algoritmy integrované v programovacím prostředí Microsoft .NET Framework 4.0, jazyk C#. Klientská aplikace i webový server byly při testu spuštěny na témže počítači, což mělo vliv pouze na kratší dobu komunikace obou stran, která ale nebyla předmětem měření. Parametry testovacího stoje byly tyto: CPU Intel Core 2 Duo 2,5GHz, RAM 4GB, HDD 7 200 RPM, OS Windows 7 64bit, IIS 7.5.

[3] SRP – Secure Remote Password – protokol pro ověřování identity na základě hesla, které ovšem není v žádné dešifrovatelné formě zasíláno mezi komunikujícími stranami. [1]

[4] salt – náhodná data přidaná k heslu před výpočtem jeho hashe; komplikuje slovníkové útoky [10 str. 390]



Obrázek 1 Diagram činností zobrazující autentizační postup

Testovací aplikace pro autentizaci používala zde popsaný autentizační mechanizmus. Unikátnost saltu byla postavena na prosté GUID hodnotě, ukládané v samostatné databázi o jedné tabulce s jedním sloupcem. Samotná funkce webové služby byla ryze testovací. Kromě autentizačních údajů přijímala jen textový řetězec, který v případě pozitivního ověření uživatele převrátila a vrátila jej v opačném pořadí („ABC“ → „CBA“).

Měřeny byly začátky jednotlivých operací pomocí třídy „Stopwatch“ v tzv. „Ticks“ (1 tick = 100 ns) a na základě rozdílů těchto začátků byla vypočtena časová rozmezí, potřebná pro vykonání dané operace. Čas na posílání dat mezi klientem a serverem byl kvůli nemožnosti existence jedněch stopek pro klienta i server vypočten jako rozdíl celkového času od okamžiku zahájení odesílání požadavku na server do okamžiku přijetí odpovědi a času, po který se prováděly veškeré operace na serveru. Operacím pro odeslání požadavku a jeho přijetí pak byla každé přiznána polovina této výsledné hodnoty. Seznam jednotlivých měřených operací je následující:

1. **Sestavení požadavku** – Vygeneruje se náhodný text o délce 32 znaků.
2. **Vytvoření ticketu** – Vygeneruje se nový salt (hexadecimální vyjádření nové GUID hodnoty), ten se spojí s hashem hesla přihlášeného uživatele a vypočte se hash této hodnoty. Výsledný hash, salt použitý v tomto hashi

a ID uživatele se spojí do jednoho textového řetězce (kompletní autentizační ticket).

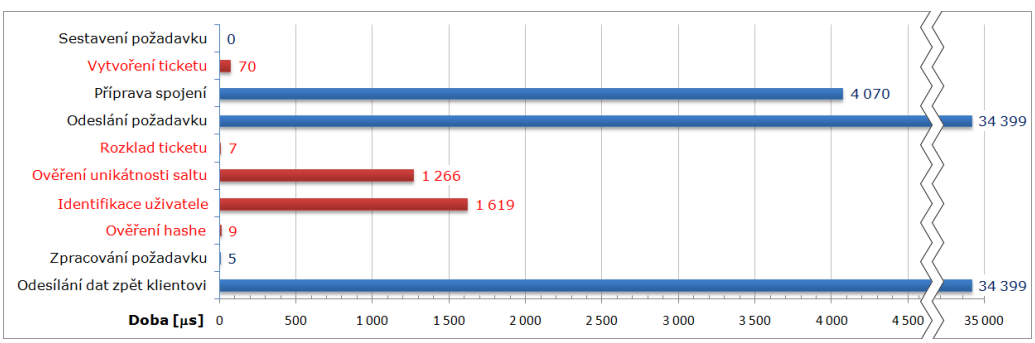
3. **Příprava spojení** – Je vytvořena instance třídy SoapClient pro volání webové služby.
4. **Odeslání požadavku** – Data jsou asynchronně odeslána webové službě, která je kompletně přijme a zahájí svou činnost.
5. **Rozklad ticketu** – Na serveru je textový řetězec (autentizační ticket) rozložen zpět na jednotlivé části (hash, salt a ID uživatele).
6. **Ověření unikátnosti saltu** – Salt je uložen do databáze saltů, přičemž kód dokáže reagovat na výjimku způsobenou pokusem o uložení duplicitní hodnoty.
7. **Identifikace uživatele** – Dle ID je v databázi nalezen uživatel a načten hash jeho hesla.
8. **Ověření hashe** – Ze spojení hashe hesla uživatele načteného z databáze a přijatého saltu je vypočten hash, který je porovnán s příchozím hashem. Výsledkem je potvrzení nebo vyvrácení identity uživatele.
9. **Zpracování požadavku** – Znaky vstupního textu jsou převráceny na opačné pořadí.
10. **Odesílání dat zpět klientovi** – Výsledný text je odeslán zpět klientovi, který jej přijme.

Test byl opakovaně (50x) proveden pro případ, kdy bylo v databázi saltů uloženo méně než 500 hodnot a pak znovu 50x, když bylo v databázi uloženo již více než 500 000 saltů, za jinak stejných podmínek. Při všech těchto pokusech byl pokaždé použit jiný vstupní řetězec (požadavek). Z naměřených hodnot byly vypočteny průměry, přičemž hodnoty jednotlivých měření se významně nelišily. Výjimku tvořilo pouze první provedení pokusu, které však bylo způsobené faktem, že ASP.NET při prvním volání provádí kompilaci webové aplikace (viz [6 str. 203]). Tyto hodnoty do výsledku samozřejmě započítány nebyly.

Naměřené hodnoty obou případů (s 500 a 500 000 salty v databázi) se však ani nyní statisticky významně nelišily, a to jak v celkových časech, tak i v rámci jednotlivých operací, včetně „Ověření unikátnosti saltu“, kde jedině bylo relevantní skutečný rozdíl očekávat. Statistická shoda hodnot naměřených v obou případech byla ověřena Mann-Whitney testem na 5% hladině významnosti. Průměr z hodnot obou pokusů ukazuje Obrázek 2.

Pouze červeně vyznačené operace se týkají procesu autentizace. Z nich je časově nejnáročnější identifikace uživatele a ověření unikátnosti saltu, což je logické, protože obě operace vyžadují připojit se k databázi a pracovat s ní. I tak ovšem kompletní zpracování požadavku trvalo pouze necelých 76 ms (75 844 μs) a z toho jen necelé **3 ms** (2 970 μs) zabraly veškeré operace pro práci s autentizačním ticketem. Zároveň při takovémto postupu není naměřený čas závislý na velikosti posílaných dat, takže se jedná o konstantní časový náklad pro libovolný, jakkoli velký datový požadavek.

Tento způsob autentizace je tedy velmi rychlý. Nutno však dodat, že v testu nebylo nijak řešeno zabezpečení posílaných dat, tj. jejich šifrování a zahrnutí kontrolního součtu do autentizačního hashe.



Obrázek 2 Průměrná časová rozpětí jednotlivých operací v testovaném pokusu

3. ZÁVĚR

Prezentovaný algoritmus umožňuje bezpečnou autentizaci pro aplikace komunikující se serverem prostřednictvím internetu. Díky vždy unikátním saltů je vyloučena útočníkem podvržená autentizace, bez znalosti správného hesla, včetně možnosti opakovaného zasílání předešlých zpráv. Princip staví na schopnosti serveru identifikovat již použité salty a striktně tak v příchozích zprávách vyžadovat jejich unikátnost.

Programátorská implementace uvedeného postupu je jednoduchá a při použití kvalitního uživatelského hesla (např. viz [7]) poskytuje komunikaci klientské aplikace se serverem dostatečně bezpečnou autentizaci, i pokud nejsou použity další zabezpečovací technologie. Přitom zpomalení komunikace, které si použitím tohoto principu každé individuální spojení vyžádá, je přijatelně nízké.

Tento autentizační postup je již úspěšně používán v praxi, a to např. v elektronickém testovacím systému představeném v článcích [8] a [9].

LITERATURA

[1] SILVA, F. O., PACHECO, J. A., ROSA, P. ***A Web Service Authentication Control System Based on SRP and SAML.*** 12th International Conference Information Visualisation. Washington DC: IEEE Computer Society, 2005. ISBN 0-7695-2409-5.

[2] HOGG, J., SMITH, D., CHONG, F., TAYLOR, D., WALL, L., SLATER, P., HOLLANDER, T., KOZACZYNSKI, W., BRADER, L., IMRAN, S. N., CIBRARO, P., DELGADO, N., CUNNINGHAM, W. ***Web Service Security: Scenarios, Patterns, and Implementation Guidance for Web Services Enhancements (WSE) 3.0.*** USA: Microsoft Press, 2006. ISBN 978-0735623149.

[3] LI, J., SUN, H., LAM, K., CHUNG, S. ***Robust Remote Authentication for ScalableWeb-Based Services.*** International Conference on Intelligent Information Hiding and Multimedia Signal Processing. Washington DC: IEEE Computer Society, 2008. ISBN 978-0-7695-3278-3.

[4] VOBORNÍK, P. ***Bezpečná autentizace aplikace klient-server v internetu pomocí povinně unikátních saltů.*** Internet, bezpečnost a konkurenceschopnost organizací 2011. Zlín: Univerzita Tomáše Bati ve Zlíně, 2011. str. 347-354. ISBN 978-80-7454-012-7.

[5] LEACH, P., MEALLING, M., SALZ, R. ***A Universally Unique IDentifier (UUID) URN Namespace.*** RFC 4122 [Online]. 2005. [cit. 10. 6. 2010] <http://www.ietf.org/rfc/rfc4122.txt>

[6] MACDONALD, M. a SZPUSZTA, M. ***ASP.NET 3.5 a C# 2008 - tvorba dynamických stránek profesionálně.*** Brno: Zoner Press, 2008. ISBN 978-80-7413-008-3.

[7] HUBÁLOVSKÝ, Š., MUSÍLEK, M. ***Počítačová bezpečnost ve výuce informatiky (Tvorba hesel a steganografie).*** Matematika-fyzika-informatika, ročník 20, listopad 2010. Praha: Prometheus, 2010. ISSN 1210-1761.

[8] VOBORNÍK, P. ***Počítačové testovací systémy.*** Sborník příspěvků z konference Alternativní metody výuky 2011. Praha: Univerzita Karlova, 2011. ISBN 978-80-7435-104-4.

[9] VOBORNÍK, P. ***Teaching algorithms using multimedia tools.*** The proceedings of the conference ERIE 2011. Praha: FEM CULS, 2011. ISSN 1803-1617.

[10] MENEZES, A., J., OORSCHOT, P., C., VANSTONE, S., A. ***Handbook of Applied Cryptography.*** Boca Raton: CRC Press, 1996. ISBN 978-0-8493-8523-0.

Ing. Petr VOBORNÍK
Univerzita Hradec Králové, Fakulta informatiky a managementu
Rokitanského 62, 500 03 Hradec Králové, Česká republika
e-mail: petr.vobornik@uhk.cz