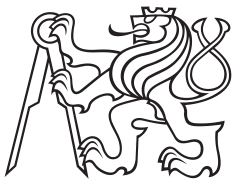
Bakalářská práce



České
vysoké
učení technické
v Praze

F7

Fakulta biomedicínského inženýrství
Katedra biomedicínské informatiky

# Vytvoření úloh a příslušných materiálů pro webové útoky

## Creating jobs and relevant materials for Web attacks

**Ondřej Pilecký**
Biomedicínská informatika

akademický rok 2016/2017
Vedoucí práce: RNDr. Dagmar Brechlerová, Ph.D.

# Poděkování / Prohlášení

Tímto chci poděkovat zejména RNDr. Dagmar Brechlerové, Ph.D. za velkou podporu při psaní této bakalářské práce. Potom bych rád poděkoval Mgr. Radimu Krupičkovi, PhD. za spolupráci při realizaci samotné bezpečnostní laboratoře. Dále chci vyřídit své díky za veškeré korektury a praktické připomínky, díky nim je tato práce čitelnější a jazykově korektní. A v neposlední řadě děkuji za stabilní podporu mé rodině, partnerce a všem dalším lidem okolo mě.

Prohlašuji, že jsem bakalářskou práci s názvem „Vytvoření úloh a příslušných materiálů pro webové útoky“ vypracoval samostatně a použil k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k diplomové práci.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

V Kladně dne 19. 5. 2017

.....

# Abstrakt / Abstract

Ve své práci jsem se zaměřil na některé webové útoky na zdravotnické informační systémy, které by mohly být použity pro výuku předmětu Bezpečnost přenosu a zpracování dat 3. ročníku oboru BMI fakulty biomedicínského inženýrství.

Tyto webové útoky jsem dále rozpracoval do jednotlivých úloh pro použití ve fakultních vyučovacích předmětech. Realizovat se budou výhradně v izolovaném prostředí na půdě univerzity a pouze s cvičnými daty.

Součástí práce je představení aplikace Safer. Na této aplikaci je možné studovat rozdíly mezi špatnou a správnou implementací zabezpečení a zároveň ho používat v praxi jako cíl těchto útoků.

Výsledkem práce je funkční aplikace, teoreticky rozpracované některé druhy webových útoků, sada úloh pro studenty univerzity procvičující dané útoky a popis stavby laboratoře, ve které lze tyto útoky bezpečně zkoušet.

**Klíčová slova:** bezpečnost, sítě, informační systém, bezpečnostní laboratoř, cvičné úkoly

In my work I have focused on several web attacks against medical information systems, which could be presented in the course Safety of Data Transmission and Processing, which is taught in the third year of the Biomedical Informatics study at the Faculty of Biomedical Engineering.

These web attacks have been worked out into individual tasks for the use in the courses which are taught at the faculty. These tasks will be realized in the isolated environment of the university campus and with test data only.

A part of the work is the presentation of the application Safer. Using this application, it is possible to study the differences between the wrong and correct implementation of the security features and to use it as the target of these attacks.

The result of this work is a released application, the description of the theory concerning several types of web attacks used in this application, a set of tasks, which can be used by the university students to practice, and the description of the building of the laboratory in which these attacks can be safely tested.

**Keywords:** security, networks, information systems, security laboratory, practise tasks

# Obsah /

# Kapitola 1
# Úvod

Téma webových útoků ve zdravotnictví je velmi ožehavé, až téměř tajemné. V mainstreamových médiích se to hemží nepolapitelnými crackery a tajemnými black-haty, kteří se v internetové síti pohybují téměř bez zábran, jednoduše získávají data a žádný systém jim nemůže odolat.

Protože je toto téma velmi odborné a technicky i technologicky náročné, jen velmi malá část populace tuší, co se skutečně na internetu a ve zdravotnických systémech obzvlášť děje. Proto se tato práce snaží představit několik základních druhů technik, které slouží k průnikům do těchto systémů, nabízí praktické cvičení určené k pochopení podstaty daného útoku a popisuje, jak by proti těmto pokusům o průnik měla vypadat správná obrana.

Tato práce teoreticky rozpracovává několik druhů útoků a posléze představuje úlohy pro studenty univerzity k praktickému nácviku. Zároveň popisuje, jak vypadá správná implementace systému, který těmto útokům umí odolávat.

Z legálního i morálního hlediska je ovšem nepřípustné zkoušet útočit na cizí či neznámé systémy. Proto tato práce představuje software jménem Safer, který je určený jako cíl těchto útoků. Safer má představovat systém používaný v malých lékařských ordinacích. Zároveň má oproti standardně používanému systému několik vylepšení, které usnadňují práci s ním i samotné útoky.

Hlavním cílem aplikace Safer je ale představit rozdíl mezi špatným a dobrým způsobem implementace jednotlivých funkčních bloků kódu, které se starají o jednotlivé funkčnosti systému.

Práce také představuje, jak vypadá stavba bezpečnostní laboratoře, kde je možné tyto webové útoky zkoušet bez rizika poškození dalších zařízení nebo problémů se zákony.

# Kapitola 2
# Cíle práce

Cíle práce vycházejí ze zadání práce a z potřeb samotné bezpečnostní laboratoře. V této práci se snažím dosáhnout těchto cílů:

- Vybrat několik nejznámějších druhů webových útoků a teoreticky je rozpracovat s přihlédnutím ke zdravotnické problematice (kapitola 5).
- Navrhnout, implementovat a nasadit systém, proti kterému by šly vyzkoušet předem definované webové útoky (kapitola 4).
- K vybraným teoretickým útokům vymyslet a napsat úlohy, podle kterých by se dalo na předem vytvořený systém zaútočit a vyzkoušet si tak v praxi, jak tyto útoky fungují (kapitola 6).
- Navrhnout strukturu bezpečnostní laboratoře, jaké přístroje by měla obsahovat, jak je rozmístit, jaké zvolit síťové zapojení, software a další nastavení (kapitola 7).

# Kapitola 3
# Vymezení pojmů

## 3.1 Obecně používané pojmy

- **Webová zranitelnost**

  Zranitelnost je definována různě.

  Společnost *Internet Engineering Task Force*[1]) ji definuje jako: *A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy*[1].

  *The Committee on National Security Systems of United States of America*[2]) zase určuje, že zranitelnost je: *Vulnerability — Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source*[2].

  *European Union Agency for Network and Information Security*[3]) vykládá zranitelnost jako: *The existence of a weakness, design, or implementation error that can lead to an unexpected, undesirable event [G.11] compromising the security of the computer system, network, application, or protocol involved*[3].

  Zranitelnosti v informatice se dále můžou dělit například na hardwarové, softwarové, síťové a další. Pro naše potřeby postačí, když si pod webovou zranitelností představíme chybu k kódu, ať už úmyslnou, nebo neúmyslnou, kterou může potenciální útočník využít ve svůj prospěch nezamýšleným způsobem.
- **Webový útok**

  Webový útok nemá žádnou pevně danou definici. Obecně se tímto rozumí jednání útočníka, který zneužívá nebo se snaží zneužít již existující zranitelnost v systému, popřípadě narušuje či poškozuje systém jiným způsobem.
- **Vektor útoku**

  Vektor útoku je v překladu směr útoku. Je to ustálený výraz pro označení metody, jakou je útok prováděn, přes jaká média, jakou technikou a v jakém časovém horizontu.

  Vektor útoku může být kupříkladu emailový spam, útok přes vyjímatelná média, skrz webový prohlížeč či skrz malware nebo jiný škodlivý software.
- **Release Candidate**

  Release Candidate je pojem ze softwarového vývoje, kde se tento výraz používá pro téměř odladěnou verzi software nebo dokumentu. Může ještě obsahovat bugy, chyby či jiné nepřesnosti, je ale už použitelná v běžném slova smyslu.
- **API**

  API je zkratka z Application Programming Interface a označuje rozhraní pro aplikaci, skrz které by aplikace měla komunikovat s ostatními aplikacemi, popřípadě s uživateli nebo správci. [4][5]

[1]) `https://www.ietf.org/`
[2]) `https://www.cnss.gov/`
[3]) `https://www.enisa.europa.eu/`

- **MVC**
  MVC je návrhový vzor, určený k rozdělení kódu na byznysovou logiku, vykreslovací šablony a řídící kód. Vzor se doporučuje středně velkým formulářovým aplikacím, které jsou zaměřeny pro více uživatelů.[6]
- **Hacker a cracker**
  Ve spojitosti s webovými útoky se často útočníci označují jako hackeři či crackeři. Bohužel se tyto dva pojmy v dnešním světě nerozlišují, či přímo zaměňují.
  *Hacker* je označení pocházející z 80. let minulého století a označovalo nadšence zběhlého v používání a znalostech slaboproudých technologií, jako byly první počítače, telefonní systémy a elektronika obecně.[7] Dnes již hacker přeneseně označuje specialistu ve svém oboru, který vůbec s elektronikou souviset nemusí, nebo inovativního, novátorského a přemýšlivého člověka, kterého baví jít ke kořenům problémů.
  Konkrétnější označení White Hat Hacker značí znalce v oblasti informatické bezpečnosti. Ten je vázán přísným etickým kodexem chránící získaná data či přístupy do cizích systémů. Často pracuje ve specializované firmě, která nabízí penetrační testovací služby. [8]
  Oproti tomu moderní výraz *Cracker* či *Black Hat Hacker* značí škodlivého útočníka, který se snaží svým jednáním prolomit či zlomit (cracknout) zabezpečení.[9][10]
- **HTTP**
  HTTP je zkratka z názvu Hypertext Transfer Protocol. Tento protokol je na internetu široce používaný a používá se pro výměnu dokumentů ve formátu HTML. Sám o sobě neumožňuje žádný způsob šifrování komunikace, proto se začalo používat zabezpečení na internetové vrstvě TCP. Pro tento šifrovaný protokol se vžilo označení HTTPS.
- **UPS**
  UPS je zkratka z anglického `Uninterruptible Power Source`. Jedná se v podstatě o chytrou baterii, která je schopná při výpadku proudu zvenčí začít proud sama do zapojeného přístroje dodávat a tím zamezit jeho nechtěnému vypnutí. Často se kombinuje s přepěťovou ochranou, kdy UPS umí zabránit poškození přístroje napěťovými špičkami, které se občas v rozvodné elektrické síti vyskytují.

## 3.2 Zákonné pojmy

- **Osobní údaj**
  Zákon definuje, že osobní údaj je: jakákoliv informace týkající se určeného nebo určitelného subjektu údajů. Subjekt údajů se považuje za určený nebo určitelný, jestliže lze subjekt údajů přímo i nepřímo identifikovat zejména na základě čísla, kódu nebo jednoho či více prvků, specifických pro jeho fyzickou, fyziologickou, psychickou, ekonomickou, kulturní nebo sociální identitu.[11]
- **Citlivý údaj**
  Zákon definuje, že citlivým údajem je: osobní údaj vypovídající o národnostním, rasovém nebo etnickém původu, politických postojích, členství v odborových organizacích, náboženství a filozofickém přesvědčení, odsouzení za trestný čin, zdravotním stavu a sexuálním životě subjektu údajů a genetický údaj subjektu údajů; citlivým údajem je také biometrický údaj, který umožňuje přímou identifikaci nebo autentizaci subjektu údajů.[12]

## 3.3 Klasifikace webových útoků

V současné době neexistuje žádná veřejně dostupná standardizace webových útoků. Tento stav pravděpodobně vychází ze samotné povahy internetu jako decentralizovaného celku. Internet jako takový neřídí žádná organizace v tom pravém slova smyslu – nejblíže k tomuto statusu má nejspíše nezisková organizace ICANN[1]), která zajišťuje například provoz kořenových serverů DNS či přidělování IP adres jednotlivým národním správcům. Ta ale ve své agendě nemá nic, co by obsáhlo klasifikaci webových útoků.

Pro webové zranitelnosti existují organizace fungující jako databáze či seznamy těchto zranitelností. Řadí se mezi mě například *Common Vulnerabilities and Exposures* společnosti MITRE[2]) či *Computer Emergency Response Team*[3]) z univerzity Carnegie Mellon v USA.

Nicméně existují společnosti, organizace či uskupení, které se snaží tento nekonzistentní stav standardizace webových útoků změnit. Jsou to většinou odborníci vzešlí ze svého oboru, kteří cítí potřebu upozornit na tato nebezpečí a nabídnout k nim adekvátní řešení.

### 3.3.1 Společnost OWASP

Jednou z nejviditelnějších organizací zabývající se bezpečností nejen webových aplikací je společnost OWASP[4]).

- **Představení společnosti**

  OWASP (Open Web Application Security Project) je projekt a komunita zabývající se bezpečností webových aplikací zahrnujíce v to rozměry lidské, procesní a technologické.

  OWASP zahájili dne 9. září 2001 Mark Curphey a Dennis Groves.

  OWASP Foundation jako organizace v USA byla založena roku 2004 s cílem podporovat infrastrukturu OWASP a projektů. OWASP je především o sdílení znalostí v oblasti bezpečnosti webových aplikací. OWASP má pouze 3 zaměstnance a funguje s velmi nízkými náklady, které jsou hrazeny z konferenčních poplatků, firemního sponzorství, bannerové reklamy, popř. v rámci některých projektů využívá různé granty.

  OWASP - Czech Republic je česká komunita hlásící se k OWASP, mezi jejíž hlavní cíle patří osvěta v oblasti bezpečnosti webových aplikací[13], (do češtiny přeloženo na[14]).

- **Působení společnosti**

  Společnost OWASP iniciuje a koordinuje řadu projektů souvisejících s webovou, síťovou i softwarovou bezpečností. Zaštiťuje dva hlavní druhy projektů: vývojářské a dokumentační.

- **Projekt OWASP Top 10**

  Jeden z nejzajímavějších a také nejužitečnějších dokumentačních projektů v rámci online bezpečnosti je OWASP Top 10. Jedná se o seznam desíti nejzávažnějších rizik webových aplikací podle konsensu mnoha bezpečnostních odborníků. Tento seznam je vydáván zhruba každé tři roky.

  Poslední finální seznam je vydaný k roku 2013[15]. Obsahuje většinu webových rizik a útoků popsaných také v této práci. Dnes nejaktuálnější je Top 10 pro rok 2017, který je zatím pouze ve verzi Release Candidate[16].

[1]) `https://www.icann.org/`
[2]) `https://cve.mitre.org/`
[3]) `http://www.cert.org/`
[4]) `https://www.owasp.org`

Tyto dokumenty jsou velmi užitečné pro rychlý náhled do problematiky moderní informatické bezpečnosti. Není to jen prostý seznam rizik, ale každé riziko útoku také rozebírá z různých pohledů nebezpečnosti, jako je vektor útočníka, jaké slabiny zabezpečení proráží, či jaké jsou technické důsledky útoku. Mimo toho také přináší přehled způsobů základní obrany, příklady použití i další odkazy, kde je možnost danou problematiku studovat dále a do větší hloubky.

- **OWASP Top 10 2017 - nové útoky**

  Stojí za povšimnutí, že ačkoliv se celý IT průmysl za čtyři roky posunul dopředu až neuvěřitelně, nový Top 10 pro rok 2017 uvádí pouze dva nové webové útoky.

  První je nedostatečně zabezpečené API a druhý je nedostatečná ochrana proti pokusům o útok, jako je například žádná obrana proti automatických skenům zranitelností, neaktualizovaný software software či nedostatečné až úplně chybějící logy.

  Všechny ostatní útoky už byly v seznamu z roku 2013, pouze se dva z nich organizačně sloučily do jedné položky. Drtivá většina dnešních palčivých problémů už je tedy dlouho známa, studována a relativně dobře popsána.

### 3.3.2 Organizace ISECOM

Organizace ISECOM[1]) vznikla v roce 2001 ve Španělsku. Její hlavní náplní je pořádání seminářů, událostí a také udělování licencí a certifikátů za jejich vlastní školení.

- **Projekt OSSTMM**

  ISECOM pravidelně vydává bezpečnostní manuál, pojmenovaný jako OSSTMM (zkratka z anglického Open Source Security Testing Methodology Manual). Tento manuál má sloužit jako návod pro operační zabezpečení a jako metodologie. Nabízí také testování již zavedených opatření a tím jejich hodnocení.[17]

- **Projekt Hacker Highschool**

  ISECOM také provozuje projekt pojmenovaný Hacker Highschool. Je zamýšlený jako začátek pro studenty středních škol v tématu informatické bezpečnosti. Na webových stránkách projektu je k dispozici řada materiálů a systém lekcí, které mohou studenta provést skrz celou problematiku.[18]

### 3.3.3 Organizace PTES

Organizace PTES[2]) je založená roku 2009 za účelem standardizace a vzdělávání v tématu penetračního testování. Na své webové stránce nabízí pravidelně aktualizovaný přehled nástrojů, postupů, zranitelností, vektorů útoků či způsobů oznamování problémů za účelem jejich opravení.[19]

### 3.3.4 Typologie kyberútoků

V současné době se podle organizace NATO Cooperative Cyber Defence Centre of Excellence[3]) webové útoky rozdělují do následujících kategorií:[20]

- **Narušení objektu útoku**

  První kategorie se označuje jako narušení. Jedná se o velmi častý typ útoku, mající za cíl objekt útoku pouze poškodit.

- **Zničení objektu útoku**

  Druhou kategorií jsou útoky, které mají za cíl zničení objektu útoku. Mezi nejznámější patří počítačový červ STUXNET. Ten byl pravděpodobně vytvořen iz-

[1]) http://www.isecom.org/
[2]) http://www.pentest-standard.org/
[3]) https://ccdcoe.org/

raelskou armádou k sabotáži jaderné elektrárny Búšehr a závodu na obohacování uranu v Natanzu, v Íránu v roce 2009.[21]

- **Špionáž**

  Nejčastější kategorií jsou však útoky mající za cíl špionáž. Mnoho států světa vydává velké prostředky na zjišťování informací od jiných států, nebo společností. V poslední době se nejslavnějším stal případ Edwarda Snowdena[22], díky kterému vyšel najevo rozsah odposlouchávání zejména evropských politiků ze strany americké tajné služby NSA.

### 3.3.5 Závažnost průniků

Hodnocením závažnosti webových zranitelností se zabývá například otevřený standard Common Vulnerability Scoring System (CVSS)[1]). Tento standard skóruje zranitelnosti od nuly do desíti podle širokého spektra faktorů, kde deset znamená nejvážnější zranitelnost.[23]

Jeden z faktorů v CVSS je i hodnocení přímého dopadu úspěšného útoku. Dopady se měří ve třech rovinách: ve ztrátě důvěrných dat, jestli může útočník převzít kontrolu nad napadeným systémem a jaký má útok dopad na dostupnost služeb.[24]

Při hodnocení nebezpečnosti webových útoků se ale musí brát v potaz většina přítomných faktorů. Útok závisí i na schopnostech útočníka či jeho nástrojích. Je tedy třeba vnímat tuto problematiku komplexně a v kontextu ostatních faktorů.

[1]) `https://www.first.org/cvss`

# Kapitola 4
# Aplikace Safer

## 4.1 Analýza požadavků

Součástí této práce je vytvoření aplikace s názvem Safer, na které by šly bezpečně a prakticky vyzkoušet některé webové útoky. Z názorných i praktických důvodů by mělo jít nezávisle na sobě jednoduše zapínat a vypínat jednotlivé zranitelnosti. K této aplikaci jsou v jiné části této práce rozpracovány jednotlivé webové útoky (kapitola 5) a návod k jejich provedení (kapitola 6).

Webové zranitelnosti, které budou v systému zahrnuty:

- MySQL Injection
- CrossSite Scripting
- CrossSite Request Forgery
- Session hijack

Aplikace Safer je příklad systému, který by mohla používat malá ordinace pro správu svých pacientů a jejich měsíčních plateb. Tento druh obecného systému (CMS, tedy Customer Management System) je náchylný k celé řadě webových útoků. V aplikaci jsou zahrnuty jen některé z nich, ale použitý druh systému dovolí v budoucnu rozšířit aplikaci o další druhy útoků.

Další požadavek je dostupnost systému přes webové rozhraní jak pro pacienty, tak pro zdravotnický personál. Měl by obsahovat důležité funkce, které by byly očekávány i od reálného systému.

Celá aplikace je psána v jazyce PHP verze 5.6. Jako databázový systém je použita relační databáze MySQL verze 5.7.

## 4.2 Funkční specifikace

### 4.2.1 Možné akce pacientů

Uživatelé se mohou volně do systému zaregistrovat se svými kontaktními údaji. Uživatelé mají možnost měnit vlastní kontaktní údaje (jméno, příjmení a telefonní číslo). Pacienti si umí zobrazit seznam vlastních plateb. Platba za každý měsíc má u sebe název tarifu, cenu, status zaplacení a tlačítko pro přístup k faktuře.

### 4.2.2 Možné akce sester a lékařů

Sestry a lékaři mají možnost zobrazit si seznam plateb všech pacientů ve své ordinaci. U každé platby mají tlačítko, které umožňuje potvrdit platbu jako zaplacenou.

### 4.2.3 Funkce systému

Systém rozlišuje uživatele na pacienty a na lékaře a sestry, kteří jsou v systému definováni jednou sadou práv. Systém generuje platby pro každého pacienta v měsíčních

cyklech od data registrace. Systém obsahuje kontaktní formulář, který umožňuje komukoli napsat zprávu na předem určený mail. Systém obsahuje ovládací panel, který umožňuje v každé části aplikace nezávisle na sobě vypínat a zapínat jednotlivé zranitelnosti.

## 4.3 Implementace

Celá aplikace je psaná podle vzoru MVC (Model-View-Controller). Tyto části aplikace jsou umístěny v odpovídajících složkách. V další složce pojmenované `js/` jsou uloženy dvě funkce implementované v jazyce JavaScript nutné pro běh programu, konkrétně pro hashování hesel. Dále aplikace obsahuje nastavení pojmenované `config.php`, soubor se styly `style.css` a v neposlední řadě soubor `index.php`, ve kterém se nachází minimalistický router, připojení k databázi a zavolání hlavního controlleru.

Router je řešen přes PHP funkci `autoloadFunction($class)`. Ten je zajímavý z hlediska přepínání bezpečných a zranitelných implementací jednotlivých tříd. Po zapnutí jednotlivých zranitelností se dozvíme, kterou třídu máme nahrát, zavoláním funkce `classesUnsafeAffection()` ze třídy `Unsafe`. Ta vrátí seznam používaných tříd a k nim přiřazené zranitelnosti v tomto tvaru:

```
[
    'Db' => ['sqli'],
    'Model' => ['xss', 'sh', 'sqli'],
    'Csrf' => ['csrf'],
    'Login' => ['sh'],
    'LogoutController' => ['sh']
];
```

Při přidávání nových útoků v budoucnu stačí upravit tuto třídu, přidat zranitelnosti k příslušným třídám a pochopitelně implementovat zranitelnou variantu používané třídy aplikace.

Zranitelné varianty tříd poznáme podle přidaného prefixu `Unsafe` a zkratky druhu zranitelnosti ke jménu souboru i třídy.

Router samotný je implementován velmi jednoduše. V případě většího rozšiřování aplikace by bylo potřeba router přepsat, aby byl více flexibilní v pojmenovávání složek se třídami pro projekt.

## 4.4 Nasazení aplikace

Pro zprovoznění aplikace je třeba:

- nakopírovat všechny soubory do cílového umístění,
- pokud je aplikace používána na subdoméně, upravit v souboru `.htaccess` direktivu `RewriteBase`,
- upravit nastavení v souboru `config.php` - zejména proměnnou `$useProduction` a konstanty `ROOT` a `EMAIL`
- nastavit správné připojení k databázi,
- pro správné fungování faktur v souboru `config.php` nastavit i údaje k připojení k aplikaci Fakturoid.

Složka s aplikací obsahuje i soubor `safer.sql`. Ten obsahuje export databáze, použitelný pro rychlý import do nového prostředí. V této databázi jsou již založeni

dva uživatelé: pacient Marián Lakatoš s fiktivním mailem `lakatos@fbmi.cz` a heslem `123456` a lékařka Jiřina Malá s mailem `mala@fbmi.cz` a heslem taktéž `123456`.

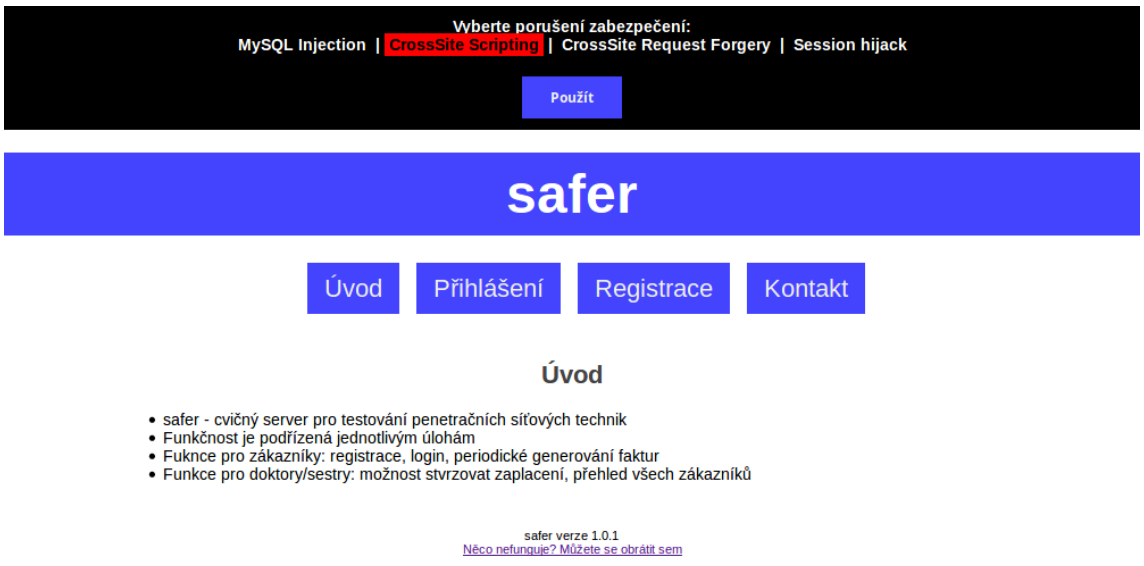
Udělování vyšších práv pro lékaře a sestry funguje buď přímou úpravou databáze vložením řádku do tabulky `admins` se správným uživatelským ID a ID ordinace, nebo přepsáním konstanty `ALLOW_MAKE_ADMIN` v souboru `config.php` a zadáním URL ve tvaru `ROOT/MakeAdmin/[userId]/[placeId]` do prohlížeče. Takto přidávat další administrátory může jen přihlášený administrátor.

## 4.5 Uživatelská dokumentace

Aplikace Safer má v horní části umístěn výrazný černý box sloužící k zapínání a vypínání jednotlivých zranitelností. Pro zapnutí stačí na zvolenou zranitelnost kliknout (ta tímto zčervená) a potom potvrdit tlačítkem `Použít`. Takto vybraná zranitelnost zůstane podbarvená červeně po celou dobu používání aplikace. Tento box je vždy dostupný při všech ostatních akcích v aplikaci.

Aplikace se ovládá zejména pomocí menu umístěného v horní části a pomocí formulářů na jednotlivých stránkách aplikace.

Na úvodní stránce je krátký přehled možností aplikace pro uživatele.




**Obrázek 4.1.** Úvodní stránka aplikace Safer se zapnutou zranitelností CrossSite Scripting

Pro registraci je třeba zadat správně všechny položky, tedy mail sloužící i jako login, vybrat tarif, heslo dlouhé minimálně 6 znaků a správně ho zopakovat. Přihlášení probíhá s daným mailem a zadaným heslem.

Po úspěšném přihlášení se můžeme kliknutím na tlačítko `Přehled plateb` dostat do sekce se seznamem všech plateb souvisejících s pacientem. Tam pacient může zkontrolovat všechny své platby, jejich status a zobrazit si příslušné faktury ve formátu PDF.


**Obrázek 4.2.** Přihlášený uživatel s pohledem na seznam plateb

Pod položkou `Změnit údaje` lze měnit své kontaktní údaje. Při změně těchto údajů je třeba správně zadat současné heslo. Toto je příklad dobré bezpečnostní praxe – u takto speciálně chráněných formulářů zabraňuje například útokům druhu CSRF.


**Obrázek 4.3.** Přihlášený uživatel s pohledem na změnu osobních údajů

Při přihlášení lékaře či sestry se v menu objeví nová položka `Pacienti`, která obsahuje seznam všech pacientů i lékařských pracovníků registrovaných v aplikaci.

V tomto zobrazení může lékař či sestra jednotlivé uživatele aktivovat či deaktivovat, což je funkce sloužící k pozastavení automatického generování nových faktur.

Systém také obsahuje funkčnost zobrazování informačních zpráv pro uživatele. Zprávy jsou trojího druhu - zprávy o úspěchu (zelené), informační zprávy (modré) a zprávy o neúspěchu (červené).


**Obrázek 4.4.** Seznam pacientů se zobrazenou úspěšnou zprávou

# Kapitola 5
# Webové útoky na informační systémy

Všechny ukázky kódu jsou vyjmuty z přiloženého systému Safer psaného v jazyce PHP a operujícího s databází MySQL.

## 5.1 SQL injection

### 5.1.1 Systémy náchylné k útoku

Náchylné k útoku jsou veškeré systémy, které obsahují databázi a zároveň nemají ošetřené uživatelské vstupy. Z těch důležitých to můžou být například informační systémy, bankovní systémy, e-shopy; z méně důležitých všechny ostatní, které uchovávají jakákoliv osobní, citlivá či jiná data.

### 5.1.2 Možnosti škod

Závažnost tohoto útoku je vysoká – tímto způsobem je možné získat data jako jsou uživatelská jména, hesla (v lepším případě „jen“ jejich hashe), vnitřní firemní informace, osobní informace jako jsou rodná čísla nebo informace o kreditních kartách.

Zneužití je nasnadě – osobní informace lze prodat, informace z oblasti finančnictví se dají zneužít přímo například přístupem na účet a převedením financí na útočníkovo konto, vnitřní firemní informace se dají cíleně prodat konkurenci.

Mimo získání dat lze tímto útokem všechna data i zničit a nebo dále pozměňovat – například v případě úspěšného útoku na e-shop útočník může pravidelně a nepozorovaně vkládat nové objednané a zaplacené zboží pro zaslání a tím ho okrádat po velmi dlouhou dobu.

### 5.1.3 Závažnost z pohledu medicíny

Největším problémem je jednoznačně únik osobních a citlivých dat o pacientech. Mimo to může útočník získávat například vstupy na zabezpečená pracoviště. Je také potřeba zmínit možnost destruktivního útoku vedeného proti nemocnici nebo jinému zařízení, které lze vymazáním všech dat (v kombinaci se zabráněním jejich rychlé obnovy ze zálohy) na několik hodin až dní efektivně vyřadit mimo provoz.

### 5.1.4 Technický popis způsobu útoku

Útok je realizován vložením útočníkem upraveného SQL příkazu přes neošetřený vstup v systému – kód se „injektuje“ (odtud název) do již stávajícího příkazu. Zde se velmi hodí technická znalost zejména:

- funkčnosti vnitřní struktury podobných systémů (například z minulé činnosti nebo v zaměstnání, kde se již s podobným systémem setkal),
- komentování kódu,
- ovládání příkazů JOIN a UNION pro přístup k celé databázi (nejen k jedné konkrétní tabulce, stejné jako pro originální příkaz),

- zkušenosti s escapeováním znaků.

Jako vstup se nejvíce používá textové pole nacházející se na webové stránce, méně často pak úprava URL adresy, manipulace s HTTP metodou POST či použití upravené a podstrčené cookie.

Rozlišujeme full SQL injection a blind SQL injection. Při full SQL injection je realizován celý proces útoku popsaný výše. Při blind SQL injection útočník nemá přístup k výsledným datům (jejich výpisu) jako takovým – může tedy efektivně provádět „jen“ zjišťování struktury databáze, vkládání nebo mazání dat.

### 5.1.5 Obrana proti útoku

Ochrana na straně aplikace je realizována například pomocí `Prepared Statements`, které sice byly vyvinuty pro optimalizaci přístupu do databáze, jako vedlejší efekt se ale ukázalo, že jsou vhodné i na kvalitní ošetření uživatelských vstupů. Dále se používá i metoda přímého escapeování znaků – typicky jsou před všechny funkční znaky u uživatelského vstupu vložena zpětná lomítka (escape znak pro SQL) a tím se eliminuje jejich funkčnost z hlediska příkazů.

Ochrana na straně databáze spočívá v omezení práv, například pro mazání záznamů z webového prostředí či omezení pohledů jen na potřebné k chodu systému. Velkou výhodou je také pravidelná a kvalitní záloha databáze v případě manipulativního (útočník provedl změny v databázi a administrátor není schopen rozpoznat jaké), či destruktivního (smazání) útoku – útoku sice nezabrání, je ale zásadní pro obnovu zasaženého systému v rychlém čase zpět.

### 5.1.6 Ukázka bezpečné implementace

Dvě funkce z třídy `Db` sloužící pro výpis dat z databáze:

```
//return all results as array of associative arrays
public static function queryAll($query, $parameters = []) {
    try {
        $result = self::$connection->prepare($query);
        $result->execute($parameters);
        return $result->fetchAll();
    } catch (PDOException $e) {
            self::reportProblem($e);
            return false;
    }
}

//method for sql manipulation
//for example for IMSERT or UPDATE
public static function queryModify($query, $parameters = []) {
    try {
        $result = self::$connection->prepare($query);
        $result->execute($parameters);
        return true;
    } catch (PDOException $e) {
        self::reportProblem($e);
        return false;
    }
}
```

## 5.2 CSRF/XSRF – Cross site request forgery

### 5.2.1 Systémy náchylné k útoku

Tento útok se zaměřuje zejména na akce již přihlášených uživatelů. Cílem je provedení konkrétní akce v systému pod autorizací oběti útoku, ideálně bez jejího vědomí. Často se označuje jako „útok pomocí zmateného zástupce“ - neošetřená stránka akci přijme i od útočníka, protože nerozezná, kterou akci si vyžádal pravý uživatel a který příkaz mu poslal útočník pouze autorizací oběti.

### 5.2.2 Možnosti škod

Tímto útokem lze realizovat prakticky jakoukoli akci, kterou může dělat přihlášený uživatel. Hodí se zejména pro falšování jinak dobře zabezpečených anket, nevědomé vkládání příspěvků na fóra, které by jinak oběť nikdy sama nenapsala, nebo nákupy v e-shopech bez vědomí uživatele (ideálně s pomocí jeho uložené bankovní karty).

Mezi nebezpečnější akce patří zejména změna hesla a emailu v samotném účtu a tím jeho reálné odcizení. Vysoká nebezpečnost je v tomto útoku pro administrátory. Když někdo odcizí účet administrátorovi systému, získá veškerá práva daného administrátora v cílovém systému. Pokud měl administrátor přístup k nastavení firewallu či gatewaye, jejich nastavení lze měnit také, což může do systému zanést další vektory útoku, které útočník může využít pro další činnost. Tímto útokem lze realizovat i průnik na uzavřený firemní intranet.

### 5.2.3 Závažnost z pohledu medicíny

Útočník při útoku na lékařský personál může jednorázově získávat léčiva na předpis, objednávat se na drahá vyšetření nebo se vyhýbat placení zdravotnických poplatků. V případě cílení například na administrátora systému nemocnice jednorázově získá plný přístup k celému nemocničnímu systému se všemi důsledky z tohoto vyplývajícími.

### 5.2.4 Technický popis způsobu útoku

Druh útoku se volí zejména podle druhu systému. Záleží na tom, jaké HTTP metody cílový systém používá pro provedení akcí uživatele.

Nejjednodušší způsob je realizace přes URL parametr GET. Při něm stačí přesvědčit oběť, aby vstoupila na stránku přes nastrčený odkaz útočníkem – jednoduchý a bohužel častý příklad je třeba zmanipulovaný odkaz na anketu.

V případě útoku metodou POST je potřeba krok navíc – útočník si na svém serveru vytvoří podvržený formulář se stejně pojmenovanými prvky a oběť se musí dostat odkazem na tento formulář – ten ji při odeslání nasměruje zpět na cílovanou stránku a odešle při tom předvyplněný POST formulář.

### 5.2.5 Maskování útoku

CSFR je v čisté podobě útok velmi očividný, proto se k němu druží celá řada pomocných technik zaměřených zejména na jeho zamaskování, aby si v ideálním případě oběť ničeho nevšimla.

Velmi rozšířené je například:

- maskování odkazů (přímo nebo s přesměrováním), dále se používá
- otevření stránky, kde útok probíhá v rámu o minimální velikosti v kombinaci s automatickým odesláním formuláře např. JavaScriptem (a k tomu další stránku s libovolnou informací, často třeba s falešnou chybou),

- přístup na odkaz s útočným skriptem pomocí externího zdroje (například pomocí pokusu o načtení externího obrázku).

Samostatnou kategorií je možnost použití v emailu vykreslující HTML formát. Velmi nebezpečná je kombinace s útokem XSS. Diskuze také vyvolává používání univerzálních přihlašovacích údajů pro velké množství služeb najednou – jakmile je prolomena jedna, jsou k tomuto útoku náchylné i všechny ostatní.

### 5.2.6 Obrana proti útoku

Technik bránění proti tomuto útoku je více, ovšem realizovatelné jsou jen na straně serveru – uživatel prakticky nemá možnost dobrý CSRF útok zaregistrovat. Pro něj lze doporučit snad jen vyhnout se automatickému přihlašování a klikání na odkazy z neověřených zdrojů, zejména v emailech, fórech a na sociálních sítích.

Na straně serveru se za obecně bezpečnou považuje ochrana pomocí autorizačního tokenu, kdy pro každý požadavek uživatele server vygeneruje token, který kontroluje při poslání požadavku od uživatele zpět na server – tento token by musel útočník při přístupu ze svých stránek uhodnout. Bohužel, v případě kombinování tohoto útoku s XSS lze tuto obranu obejít.

Bezpečnější způsob, který je méně náchylný na XSS, je ovšem nepoměrně složitější na implementaci a zdroje a proto málo používaný, je obecně označovaný jako `ticket system`. Ten spočívá ve vydávání ticketů pro každý konkrétní požadavek od uživatele. V případě poslání požadavku zpět ho server pomocí vydaného ticketu zkontroluje, zda byl vydán pro tento konkrétní požadavek a všechny ostatní akce vyhodnotí jako CSRF.

Jako další metodu je možno použít náhodná jména pro políčka samotného formuláře, které by opět útočník musel uhodnout pro každou akci.

### 5.2.7 Ukázka bezpečné implementace

Třída `Csrf` řešící vydávání a následnou kontrolu tokenů se zabudovanou kontrolou stránky, ze které uživatel akci spouští:

```
<?php
class Csrf extends Model {
    public static function getCsrfToken () {
        $csrfToken = self::getRandomHash();
        //added extra layer with adding actual uri into hash
        $actualUri = 'http://' . $_SERVER['HTTP_HOST'].
            $_SERVER['REQUEST_URI'];
        $csrfRefererToken = hash('sha512', $actualUri.
            $csrfToken, false);
        Db::queryModify('INSERT INTO `csrf` (`user_id`, `token`,
            `active`, `timestamp`) VALUES (?, ?, 1, NOW())',
            [$_SESSION['id_user'], $csrfRefererToken]);

        return $csrfToken;
    }

    public static function validateCsrfRequest($returnedToken) {
        $storedToken = Db::querySingleOne('SELECT `token` FROM `csrf`
            WHERE `user_id` = ? AND `active` = 1
            ORDER BY `id` DESC', [_SESSION['id_user']]);
        //unactive all entries
```

```
            Db::queryModify('UPDATE `csrf` SET `active` = 0
                WHERE `user_id` = ? AND `active` = 1',
                [$_SESSION['id_user']]);
            //add referer uri into hash to get stored value
            $returnedRefererToken = hash('sha512', $_SERVER['HTTP_REFERER'].
                $returnedToken, false);
            if ($storedToken != $returnedRefererToken) {
                self::newTicket('warning', $_SESSION['id_user'],
                    'CSRF attack! Token: '.$storedToken);
                return false;
            } else {
                return true;
            }
        }
    }
```

## 5.3 XSS – Cross-site scripting

### 5.3.1 Systémy náchylné k útoku

Systémy s neošetřenými vstupy a výstupy proti jazyku Javascript.

### 5.3.2 Možnosti škod

Škodlivý kód lze použít nejčastěji pro:

- získání citlivých údajů návštěvníků stránek,
- při phisingu pro vložení pozměněného/jiného obsahu na jinak důvěryhodnou stránku,
- pro změnu vzhledu stránky (nevyžádaná reklama, snížení důvěryhodnosti stránky),
- pro její přímé znefunkčnění,
- kvůli odcizení PHP session indentifikátoru (viz. útok Session hijack), nebo obcházení bezpečnostních kontrol, které typicky slouží jako příprava pro další druhy útoků.

### 5.3.3 Závažnost z pohledu medicíny

V případě webového medicínského systému lze takto získat kontrolu nad lékařským účtem a tedy provádět všechny akce jako přihlášený lékař včetně procházení citlivé dokumentace pacientů. Mimo to může sloužit jako účinná příprava phisingu či jiných druhů útoků.

### 5.3.4 Technický popis způsobu útoku

Útoky XSS se většinou dělí na tři typy.

- První typ se nazývá „local“. Útok je realizován přes úpravu odkazu, typicky přes neošetřený parametr, který se přímo vypisuje.
- Druhý typ je označován jako „reflected“ – pokud stránka přejímá pomocí např. jazyka PHP text a dále ho interpretuje na stránku a tento vstup není ošetřený, může ho útočník využít ve svůj prospěch.
- Nejzajímavější a nejsilnější pro útok je třetí typ označovaný jako „peresistent“, který stránku modifikuje trvale. Pro úspěch je třeba kód útočníka dostat ve správném tvaru do databáze a nechat ho vypsat přes neošetřený výstup na webové

stránce – tento kód totiž spustí každý (a tím se stane terčem útoku), kdo si nechá načíst a vypsat tento neošetřený výstup.

### 5.3.5 Obrana proti útoku

Ze strany uživatele není moc možností, jak tomuto útoku zabránit. Jediná spolehlivá možnost je vypnout celý Javascipt, čímž se ale práce pro uživatele obecně výrazně znepříjemní až znefunkční.

Ochranu je třeba provádět hlavně na straně serveru. Při vkládání nebo vypisování dynamického obsahu (z uživatelského vstupu, z databáze apod.) je nutno každý text ošetřit převedením speciálních znaků na HTML entity (například ze znaku `>` vytvořit řetězec `&gt`), které se vykreslí stejně, nespustí ale žádný funkční kód.

Jsou tu možné dva přístupy: přísná kontrola při vkládání dat do systému, nebo ošetřování až při jejich vypisování. Pochopitelně je možné oba přístupy kombinovat pro zvýšení zabezpečení.

### 5.3.6 Ukázka bezpečné implementace

Funkce `sanitize` z třídy `Model`, která zodpovídá za ošetření výstupu aplikace do jazyka HTML:

```
public function sanitize($data) {
    if ($data == null) {
        return null;
    } elseif (is_string($data)) {
        return htmlspecialchars($data, ENT_QUOTES);
    } elseif (is_array($data)) {
        foreach ($data as $key => $value) {
            $data[$key] = $this->sanitize($value);
        }
        return $data;
    } else {
        return $data;
    }
}
```

Klíčové je použití funkce `htmlspecialchars` se zapnutou direktivou `ENT_QUOTES` pro ošetření jednoduchých uvozovek, které funkce v základním použití neošetřuje.

## 5.4 Session hijack – únos spojení

### 5.4.1 Systémy náchylné k útoku

Veškeré systémy s přihlašováním – používá se v kombinaci s XSS nebo CSRF i v přes internet nepřístupných intranetech.

### 5.4.2 Možnosti škod

Tímto útokem se dá efektivně převzít kontrola nad účtem již přihlášeného uživatele. Útočník tedy může provádět všechny akce jako oběť. Cílem jsou zejména administrátoři, uživatelé zabezpečených systémů nebo zákazníci v e-shopech, kteří mají v daném obchodě uložené kreditní karty či předkoupený kredit.

### 5.4.3 Závažnost z pohledu medicíny

Nebezpečnost tkví zejména v úniku citlivých a osobních informací o pacientech, ke kterým mají přístup pouze ošetřující lékaři, a v převzetí kontroly nad medicínským systémem v případě úspěšného útoku na účet administrátora.

### 5.4.4 Technický popis způsobu útoku

Útok se realizuje pomocí odcizení identifikátoru (ve většině případů `session cookie` uložené v prohlížeči), který webové systémy používají k jednoznačnému označení aktuálně přihlášeného uživatele. Toto lze provést různými způsoby.

- Prvním způsobem je prostá fyzická krádež zmíněného alfanumerického řetězce – zde je ale potřeba fyzický nebo plný virtuální přístup ke stroji oběti.
- Druhý je nazývaný `session fixation` a spočívá v podstrčení předvolené session ID oběti, které jsou potom přiřazena práva oběti (a s tímto podstrčeným session ID může stále pracovat i útočník).
- Třetí technika spočívá v získání session cookie pomocí výše popsaného útoku XSS.
- Čtvrtý způsob je nazývaný „session sidejacking“, při kterém útočník odposlouchává síťový provoz. Pokud server a klient nepoužívají HTTPS nebo jiný šifrovací protokol, session ID je v komunikaci dohledatelná v čisté podobě, takže útočník si ji může jednoduše zkopírovat – tímto se dá obejít praktika některých serverů, které pro přihlašování sice používají protokol HTTPS, pro další provoz už ale jen HTTP, zřejmě z důvodů jednodušší implementace a menší zátěže na serveru. K tomuto druhu útoku jsou navíc velmi náchylné veřejné wifi hotspoty, protože tam může odposlouchávat síťový provoz kdokoliv bez žádného dalšího omezení.

### 5.4.5 Obrana proti útoku

Obrana vyplývá ze způsobů odchycení session ID.

- První důležitá věc je neodcházet od nezamčeného počítače a tím zamezit jeho neoprávněnému použití v případě nepřítomnosti.
- Dále je důležité neklikat na webové odkazy, pokud nepocházejí v důvěryhodného zdroje, obzvláště v emailech, které lze snadno podvrhnout.
- A v neposlední řadě je třeba vyhnout se pokud možno zcela použití veřejných wifi hotspotů. Když to není možné, alespoň se nepřihlašovat do žádných webových služeb, zejména do těch postrádajících šifrování.

### 5.4.6 Ukázka bezpečné implementace

Část knihovny `Model` určená pro kontrolu platnosti přihlášení:

```
public function checkLogin() {
    if (!isset($_SESSION['username'], $_SESSION['login_string'])) {
        return false;
    }

    $DBpassword = Db::queryOne('SELECT `password` FROM `users`
        WHERE email = ?', [$_SESSION['username']]);
    if ($DBpassword[0] == null) {
        return false;
    }
```

```
    $passwordCheck = hash('sha512', $DBpassword['password'].
        $_SERVER['HTTP_USER_AGENT']);
    if ($passwordCheck != $_SESSION['login_string']) {
        return false;
    }

    //success
    return true;
}
```

Část funkce `tryLogin` třídy `Login`, zodpovědné za přihlášení a tvorbu kontrolních elementů

```
class Login extends Model {
public function tryLogin($data) {
    $attempt = Db::queryOne('SELECT `id_user`,`email`,`password`,`salt`
        FROM `users` WHERE `email` = ?', [$data['login']]);
    $userPassword = hash('sha512', $data['p'].$attempt['salt']);

    //if user doesn't exists
    if ($attempt == null) {
        return ['error', 'Bohužel, uživatel není v databázi.
            <a href="'.ROOT.'/registration">
                Nechcete se raději registrovat?
            </a><br/>'];
    }

    //account is not locked
    if ($this->checkBrute($data['login']) == false) {
        //password is different!
        if ($userPassword != $attempt['password']) {
        //write it into brutcheck
        Db::queryModify('INSERT INTO `login_attempts`
            (`login`,`success`,`timestamp`)
            VALUES (?, 0, NOW())', [$data['login']]);
        return ['error', 'Bohužel, heslo není správně.
            <a href="'.ROOT.'/GetLinkForNewPassword">
                Nepotřebuješ si nechat zaslat nové?
            </a><br/>'];
        } else {
            //corrent both login and password - success!
            //store information about newly logged user
            $_SESSION['id_user'] = $attempt['id_user'];
            $_SESSION['username'] = $data['login'];
            $_SESSION['login_string'] = hash('sha512',
                $userPassword.$_SERVER['HTTP_USER_AGENT']);
            Db::queryModify('INSERT INTO `login_attempts`
                (`login`,`success`,`timestamp`)
                VALUES (?, 1, NOW())', [$data['login']]);
            return ['success', 'Přihlášeno, vítejte zpět!'];
        }
    } else { //account is locked by bruteforce, don't allow login }
}
```

# Kapitola 6
# Scénáře webových útoků

Scénáře s návody k webovým útokum jsou určené pro výuku příslušného předmětu na fakultě FBMI. Všechny operace by se měly odehrávat na izolované, k tomu určené síti. Tréninky útoků probíhají proti systému Safer, který je k tomu účelu navržený, a je dále rozepsaný v kapitole 4.

## 6.1 SQL injection

### 6.1.1 Nastavení a motivace

Motivací útočníka bude potvrdit si zaplacení platby bez vědomí pověřeného pracovníka. V tomto druhu útoku pracujeme pouze se samotným systémem.

### 6.1.2 Postup

Krokem číslo jedna je zjištění struktury té části aplikace, která nám umožní zaplatit platbu. Zajímat se budeme hlavně o tabulku ‘admins‘, ve které se drží záznamy o administrátorech na jednotlivých pracovištích. Tabulka má záznam o uživateli a místu, kde je uživatel administrátorem. Jakmile se nám povede zapsat se do této tabulky, můžeme si platbu potvrdit pohodlně sami pomocí dostupného webového rozhraní.

Krokem číslo dva je zjištění dat, která budeme chtít vložit. ID uživatele je jednoduše zjistitelné z adresního řádku při kontrole plateb, těžší je zjistit ID místa, kde jsme zapsaní. Můžeme hádat a s trochou štěstí jej i uhodnout, nicméně pokud se podíváme na registrační formulář, kde se místa s tarify vypisují, můžeme ze zdrojových kódů vyčíst (a nebo si přímo tipnout), že se místa s tarify budou vypisovat podle ID, tedy že ID požadovaného místa můžeme odpočítat odshora podle pořadí v seznamu.

Krok číslo tři je vytipování slabého místa v aplikaci – nejvhodnější je zřejmě formulář nacházející se na adrese `/changePersonals/[ID uživatele]`, který na první pohled umožňuje přímý zápis do databáze. Je ale možné použít i formulář pro přihlášení nebo registraci, aplikace je nezabezpečená na všech vstupech.

Krok číslo čtyři: samotná realizace útoku. Vlastní zápis do databáze se realizuje například odesláním tohoto řetězce v posledním textovém poli:

```
moje_cislo" WHERE 0=1;
INSERT INTO admins (user_id, place_id) VALUES
([ID uživatele], [místo kde je uživatel zapsaný]); --
```

První uvozovkou se uzavře řetězec označující telefonní číslo (kdyby nefungovalo, můžeme zkusit uzavřít uvozovkou jednoduchou); následuje `WHERE` příkaz a neplatná podmínka abychom původní příkaz vyhodnotili vždy jako neplatný, a tím se neprovedl. Následuje středník pro uzavření celého příkazu a kompletní nový příkaz (jednoduché přidání řádku do tabulky `‘admins‘`). Ukončíme opět středníkem pro konec příkazu a následuje komentář s následující mezerou, která je pro syntax komentáře v MySQL

nutná (zde je možné použít v případě nefunkčnosti i neukončený komentář pro více řádků). Nyní jsme se stali z pohledu systému administrátory, můžeme tedy pohodlně kliknout na tlačítko `Zaplatit` ve webovém rozhraní a tím si potvrdit svou platbu.

Krok číslo pět je nepovinný, ale vhodný – měli bychom po sobě zamést stopy, jinak by si správce systému mohl všimnout, že bez jeho vědomí přibyl nový administrátor. Krok je realizovaný stejnou cestou jako vložení záznamu, jen ho musíme zase smazat. Mohl by k tomu sloužit například řetězec:

```
moje_cislo" WHERE 0=1;
DELETE FROM admins WHERE user_id = [ID uživatele]; --
```

s velmi podobnou syntaxí jako příkaz pro vložení řádku.

## 6.2 CSRF/XSRF – Cross site request forgery

### 6.2.1 Nastavení a motivace

Pro scénář předpokládáme dostupnost zdrojových kódů pro útočníka, neodhlašování se oběti (zde zdravotnické sestry) ze systému a podstrčení zmanipulovaného odkazu, na který oběť klikne.

Útočník (v tomto případě zaregistrovaný pacient) má motivaci nechat si zapsat do systému platbu, kterou sestra vědomě nikdy nepotvrdila. Z pohledu systému to bude oprávněná akce, z pohledu účetnictví budou peníze sice chybět, systém ale neukáže, který pacient nezaplatil.

### 6.2.2 Postup

Provedení spočívá v nalezení funkce volané na adrese `/PayInvoice/[ID platby]`, která zajišťuje potvrzení platby do systému. První úkol (najít číslo faktury) není problém; zobrazuje se ve výpisu plateb (v případě jeho nevypisování by šel realizovat útok na všechny platby, dokud bychom netrefili tu, kterou požadujeme – tím by ovšem klesla nenápadnost celého útoku).

Při vyzkoušení ale narazíme – funkce je pro pacienty blokovaná, dostupná je pouze pro personál. Musíme tedy podstrčit přihlášené sestře zmanipulovaný odkaz. Zde už hraje roli kreativita – odkaz je možné předat v čisté podobě nebo v HTML atributu `href` tagu `<a>` s jiným popiskem. Tato řešení mají nevýhodu ve faktickém ukázání výsledku oběti a tím i jeho silně pravděpodobném prozrazení (oběť systém zná a ví, co se stalo).

Další možností je zavolat tento odkaz na jiné webové stránce pod útočníkovou kontrolou. Falešná stránka může zobrazit například error #404 a oběť zklamaně z webu odejde, netušíc, že se provedl faktický útok v průhledném okně nebo v rámu o rozměru 1px, zakomponovaném do webové stránky někde mimo zorné pole uživatele. Nevýhoda tohoto řešení spočívá v nutnosti provozovat vlastní web (nebo mít cizí web pod svou kontrolou).

Jednodušší řešení leží v předání odkazu v atributu `source` v HTML tagu `<img>` – při pokusu načíst obrázek (a tím faktického potvrzení platby) se sice neukáže žádaný obrázek, ale toho si uživatel nemusí vůbec všimnout, obzvlášť když žádný obrázek neočekává. Je ale třeba počítat s tím, že některé systémy ale načítání externích obrázků zakazují (například maily označené jako spam).

## 6.3 XSS – Cross-site scripting

### 6.3.1 Nastavení a motivace

Útok XSS je ve své podstatě velmi jednoduchý, a zároveň velmi mocný. Jeho správné použití vyžaduje širokou znalost jazyku JavaScript, který je ve všech moderních prohlížečích standardně dostupný.

Pro jednoduchost zde ukáži jednoduchou demonstraci jeho použití: vyvoláme si v prohlížeči vyskakovací okno.

### 6.3.2 Postup

Nejdříve musíme najít vhodné umístění pro útok. Pro největší účinek (tedy trvalé, perezistentní XSS) je třeba, aby se útočný řetězec uložil. Při projití aplikace zjistíme, že můžeme volně editovat údaje u pacienta, a ty se poté zobrazují v seznamu všech pacientů všem administrátorům - toto bývá tradiční místo pro tento druh útoku.

Pro provedení útoku stačí vložit na nesprávně ošetřené políčko jména či telefonního čísla řetězec obsahující JavaScript. Běžným způsobem demonstrace bývá zobrazení modálního okna s číslovkou jedna - tohoto dosáhneme vložením řetězce `<script>alert('1');</script>`.

Po vložení řetězce stačí počkat, až si administrátor systému zobrazí souhrnný seznam informací o pacientech a jako první se provede skript, předem vložený námi přes kontaktní údaje.

## 6.4 Session hijack – únos spojení

### 6.4.1 Nastavení a motivace

Předpokládáme použití systému na síti nezabezpečené proti odposlechu a nepoužití šifrování v síťové komunikaci. Dále je potřeba aktivní komunikace oběti (zde ordinujícího lékaře) se systémem nacházejícím se na druhém konci sítě.

Motivací útočníka je získat práva administrátora systému – kupříkladu pro čerpání volných služeb vedených v systému, vykrádání cizích dat nebo přeprodej dalších služeb pacientům, realizovaných pomocí ukradeného účtu.

### 6.4.2 Postup

Prvním krokem je samotné připojení se na síť – realizované pomocí wifi v lékařské ordinace či fyzickému připojení do volné zásuvky v ordinaci.

Druhým krokem je odchycení `cookie`, které posílá lékařský počítač komunikující se systémem například programem Wireshark nainstalovaným v operačním systému Kali Linux. Síťový provoz je třeba vyfiltrovat; po prostudování struktury posílaných cookies se nabízí filtrovat jen cookies s názvem `login_string` a dohledávat `id_user` a `username`, nebo je například možné filtrovat jen dotazy s cookie, které obsahují ID lékařského účtu (pokud ho známe).

Třetím krokem je přenos odposlechnutých cookies do prohlížeče a faktické zneužití účtu – toto je možno realizovat například zabudovanými funkcemi Import cookies a Export cookies, nebo pluginy do browserů implementující tyto funkce dodatečně.

# Kapitola 7
# Bezpečnostní laboratoř

## 7.1 Zadání

Laboratoř má poskytnout fyzický prostor pro školení určené pro lékaře a zdravotnický personál všech specializací a odborníky v nelékařských zdravotnických oborech. Úkolem této bakalářské práce je připravit podklady pro realizaci této laboratoře.

### 7.1.1 Praktická část školení

Zadání praktické části školení lékařů v dané laboratoři obsahuje úlohy pro:

- **autentizaci** - různé metody autentizace, heslo, ověření síly hesla, politika hesel, více hesel, změna hesla, bezpečné ukládání hesel, mazání, ukládání a archivace dat,
- **data** - bezpečný přenos dat, digitální podpis, šifrování, anonymizace, bezpečná emailová komunikace,
- **digitální stopu na internetu** - sledování počítače, informace v dokumentech a v komunikaci,
- **dezpečnost a IT hrozby** - zabezpečení počítače, napadení a ochrana mobilních zařízení, viry, antiviry.

Tyto všechny úlohy a scénáře útoků musí být na navrženém schématu realizovatelné.

### 7.1.2 Požadavky pro provoz a správu

Celá laboratoř musí být síťově oddělena od všech ostatních sítí v okolí. Nesmí být možné se dostat ze sítě bezpečnostní laboratoře.

Zároveň je ale třeba splnit požadavek vzdálené správy celého systému. Mělo by být možné se vzdáleně k celé bezpečností laboratoři přihlásit a mít možnost ji na dálku administrátorsky spravovat, popřípadě ji celou vypnout.

## 7.2 Realizace hardware

Ke splnění podmínek realizace bude potřeba následující hardware:

### 7.2.1 Server

Serverem myslíme stroj v konfiguraci:

- procesor: Intel Xeon generace Haswell, 4 jádra, 3,1 GHz,
- operační paměť: DDR3, 4 GB s možným rozšířením na max. 32 GB,
- pevné úložiště: WesternDigital5000LUCT, velikost 2,5 palce, úložiště 500GB,
- další parametry: uzpůsobený pro zapojení do racku, 2 síťové karty, bez optické mechaniky, 4 pozice celkem pro SATA/SAS disky

Server v této konfiguraci zvládne obsloužit všechny síťové požadavky bezpečnostní laboratoře a vyhovuje i požadavkům dvou oddělených síťových přístupů, zdrojům fakulty a fyzických rozměrů.

Pro jeho ochranu před napěťovými špičkami z nechráněné rozvodné sítě jsme použili UPS značky Eaton typu 5E 650i USB. Tato UPS navíc zabrání nechtěnému náhlému vypnutí serveru, který by se tím mohl poškodit. UPS lze také nakonfigurovat tak, aby při výpadku proudu zvenčí vyslal serveru signál, a ten se sám bezpečně vypne a zabrání tím opět nechtěnému vypnutí po přerušení dodávky elektřiny ze samotné UPS.

Pro fyzické umístění serveru jsme použili standardní malou rackovou skříň velikosti 19 palců, kam se vešly i router, switch i UPS včetně většiny kabeláže.

Softwarová výbava serveru je rozepsána v sekci 7.4.1.

### 7.2.2 Cílové počítače

Stroje, na kterých budou pracovat účastníci školení, jsou v konfiguraci:

- procesor: Intel Core i5 generace Skylake, 4 jádra, 2,7 GHz,
- operační paměť: DDR4, 8 GB, 2133 MHz,
- pevné úložiště: 1 TB pevný disk, 7200 rpm,
- další možnosti úložišť: DVD mechanika, čtečka SD karet,
- příslušenství: monitor, klávesnice, myš.

Tyto stoje jsou ve očekávané konfiguraci dnešní doby a odpovídají požadavkům obvyklých středně výkonných kancelářských stanic. Pro účely školení v bezpečnostní laboratoři plně vyhovují.

### 7.2.3 Síťové prvky

Mezi síťové prvky počítáme zejména switch Cisco SG200-08 Smart Switch s 8 porty pro zapojení a maximální rychlostí sítě 1 Gbs. Router je ASUS RT-N12 D1 s podporou sítí IEEE 802.11 b/g/n, který pracuje v pásmu 2,4 GHz pracující s maximální rychlostí 100Mbs.

Dále také použijeme router zvládající průměrnou síťovou zátěž s bezdrátovou technologií, který umožní do sítě zapojit na dálku počítač navíc.

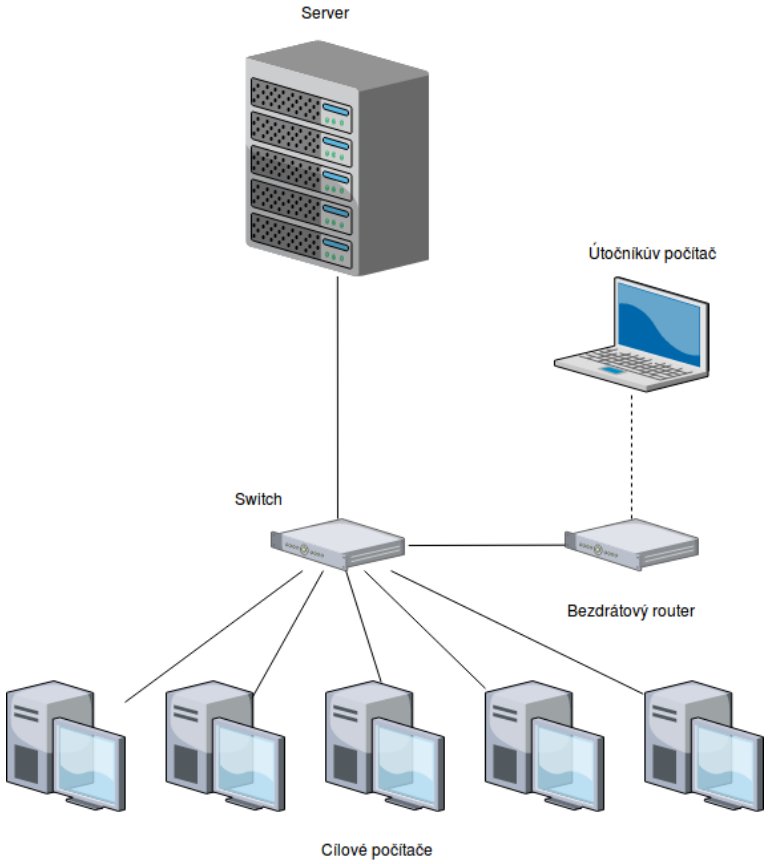
Na tyto prvky nejsou kladeny žádné vyšší hardwarové požadavky, a proto plně vyhovují potřebám bezpečností laboratoře.

## 7.3 Realizace sítě

### 7.3.1 Přehled

Uvedené zapojení musí vyhovět jak vzdálené správě celé laboratoře zvenku, tak kompletní izolaci sítě.

Schématický nákres síťového zapojení je vidět na obrázku 7.1.




**Obrázek 7.1.** Schéma sítě pro bezpečností laboratoř

### 7.3.2 Server

Hlavní součástí sítě bezpečnostní laboratoře je server. Ten je připojen jedním síťovým připojením do vnější sítě (internetu) skrz fakultní počítačovou síť. Toto spojení slouží k vnějšímu administrátorskému přístupu k bezpečnostní laboratoři. Druhé síťové připojení je připojeno do switche.

### 7.3.3 Cílové počítače

Cílové počítače musí být schopné se spojit se serverem. K tomuto doporučuji použít pevné IP adresy, které umožní větší kontrolu nad zařízeními i při jednotlivých úlohách, kde se může IP adresami manipulovat nestandardně. Zároveň to umožní mít obrazy jednotlivých strojů s přesnou síťovou konfigurací a použít je k přímé obnově například softwarově poškozeného či jinak nefunkčního počítače.

V případě potřeby je možné na serveru použít službu DHCP, která by IP adresy přidělovala. Usnadní se tím konfigurace a počáteční nastavení, ztratíme ale kontrolu nad přesným umístěním jednotlivých strojů v síti a obnova počítačů nebude tak exaktní – nevyřešíme tím tedy například špatnou konfiguraci v síťovém nastavení způsobenou nesprávným použitím vybavení laboratoře.

### 7.3.4 Další síťové prvky

Do switche je zapojený server a také cílové počítače, které tímto mohou přes switch komunikovat se serverem.

Do téhož switche je také zapojený bezdrátový router sloužící pro přístup dalším počítačům, například notebooku útočníka. Konfigurace tohoto routeru je popsaná v sekci 7.4.5.

## 7.4 Softwarové nastavení všech prvků laboratoře

### 7.4.1 Fyzický server

Fyzický server musí v laboratoři plnit dvě funkce. Za prvé musí zabezpečit samotnou realizaci úkolů, tedy obsahovat software, na který se bude reálně útočit. Za druhé musí obsloužit distribuci správných souborů k použití do cílových počítačů (v našem případě obrazů disku). Zároveň musíme pamatovat na požadavek obsluhy serveru bezpečně zvenčí. Proto bude server řešen jako dva virtuální servery s odlišnými hlavními funkcemi ve virtualizovaném prostředí běžícím na fyzickém serveru.

Pro virtualizaci ostatních serverových instancí použijeme Hyper-V[1]) od firmy Microsoft[2]). Jeho základní verze je pro fakultní použití zdarma[25]. Hyper-V nám umožní přístup na virtualizační platformu zvenčí pomocí vzdálené správy přes předem nakonfigurované síťové připojení. Díky tomuto přístupu můžeme celý server ovládat se zachováním izolace ostatních prvků laboratoře.

V dalším textu budu označovat virtuální server, starající se o samotný provoz úloh v laboratoři, jako produkční server. Druhý virtuální server, určený pro distribuci obrazů disku, budu nazývat distribučním serverem.

### 7.4.2 Produkční virtuální server

Produkční server je třeba přizpůsobit aplikaci, na které budou vedeny jednotlivé útoky. Vzhledem k tomu, že tato aplikace vyžaduje ke svému běhu programovací jazyk PHP a databázi MySQL, rozhodl jsem se použít pro server jako operační systém linuxovou distribuci Debian.

Tento operační systém s danými požadavky dobře spolupracuje a předpokládám tedy při jeho použití minimální technickou nákladnost. Díky široké používanosti tohoto operačního systému je možnost v případě problémů využít rozsáhlé databáze a dalších podpůrných materiálů.

### 7.4.3 Distribuční virtuální server

Pro distribuční server jsme zvolili taktéž linuxovou distribuci Debian. Server má z pohledu laboratoře přístupných několik obrazů disku, které budou potom sloužit v cílových počítačích pro realizaci jednotlivých úloh (pro každou jednu úlohu jeden obraz). Obrazy budou přístupné pouze ke čtení. Jejich aktualizaci či výměnu musí realizovat správce serveru přes Hyper-V.

Výhodou bude snadná správa dvou stejných operačních systémů se všemi výhodami z toho plynoucími.

### 7.4.4 Cílové počítače

Cílové počítače jsou vybaveny jako standardní kancelářské či fakultní stroje. Abychom zachovali určitou modularitu místnosti a mohli ji používat i k jiným účelům než jen jako bezpečností laboratoř, rozhodli jsme se s vedoucími práce jít cestou virtualizace.

[1]) `https://blogs.technet.microsoft.com/technetczsk/p/microsoft-hyper-v/`
[2]) `https://www.microsoft.com/cs-cz/`

Každý počítač má vlastní nezávislý operační systém. Pro laboratoř je ale důležité, že má také každý nainstalovaný program VirtualBox[1]) od firmy Oracle[2]). Všechny praktické úlohy budou probíhat přes tento program, který umožní spustit jednotlivé úlohy v samostatném odděleném prostředí (takzvaném sandboxu) a po ukončení zanechat počítač připravený k dalšímu použití a v pořádku pro dalšího uživatele.

### 7.4.5 Bezdrátový router

Router je třeba nakonfigurovat tak, aby bez konfliktů bezdrátově zapojoval počítače, které do sítě patří.

Tohoto požadavku je možné dosáhnout zapnutím služby DHCP, která bude přidělovat předem daný blok IP adres, který nebude konfliktní se zbytkem sítě.

Základního zamezení přístupů nechtěných počítačů dosáhneme zapnutím vyžadování hesla či certifikátu podle standardu WPA/2. V případě potřeby můžeme na routeru aplikovat filter povolených MAC adres, abychom tím zabránili přístupu nezamýšlených počítačů do sítě laboratoře a tím zvýšili její izolaci.

## 7.5 Praktické použití

### 7.5.1 Příprava laboratoře před školením

Před školením je potřeba celou laboratoř připravit. Bude zejména potřeba nahrát aktuální obrazy disků s požadovanými úlohami do cílových počítačů. Tohoto lze dosáhnout dvěma různými způsoby.

První možnost je manuální: před samotnou úlohou bude potřeba požádat všechny účastníky, aby každý na svém počítači stáhl ze serveru správný obraz nebo obrazy disků a uložili je. Druhá možnost je tento proces automatizovat například skriptem, který se vždy po spuštění počítačů nebo na manuální vyžádání spustí a který sám zajistí stažení požadovaných souborů.

Dále bude potřeba vždy zkontrolovat funkčnost serverů. Funkčnost distribučního serveru se pozná podle úspěchu či neúspěchu stažení nových obrazů disku, funkčnost produkčního serveru doporučuji vyzkoušet manuálně. V případě požadavku na automatizaci může být v lektorském počítači připravený skript, který funkčnost otestuje a ohlásí výsledek automaticky.

### 7.5.2 Průběh školení

Účastníci si na začátku školení zapnou program VirtualBox a pomocí něho nahrají předem stažený obraz disku. Tím se dostanou do virtualizovaného prostředí určeného výhradně pro práci s úlohou. Při použití funkce zobrazení na celou obrazovku a schování všech ovládacích prvků programu VirtualBox budou mít účastníci pocit, že nejsou nijak virtualizací omezováni v práci s počítačem.

### 7.5.3 Řešení nestandardních situací

V případě nehody při práci s počítačem nebo jiné nepředpokládané variantě můžou uživatelé celou práci s virtuálním systémem ukončit a problémový obraz disku smazat. Poté si jednoduše stáhnou novou kopii obrazu a stejným postupem jako na začátku ho spustí a můžou pokračovat v práci s úlohou.

[1]) `https://www.virtualbox.org/`
[2]) `https://www.oracle.com/cz/`

# Kapitola 8
# Diskuze

Téma webových útoků je nedílnou součástí internetu už od jeho počátku. Nelze jej od něj oddělit, nelze jej zakázat ani je úplně potlačit.

Ve své práci jsem ukázal, že stejně jako internet, webové útoky nemají přesnou klasifikaci ani danou strukturu. Svojí podstatou vycházejí z programátorských chyb a opomenutí. Naproti tomu konkrétní webové zranitelnosti jsou dobře zdokumentované, vysvětlené a hlavně informace o nich jsou snadno a zdarma dostupné. Mnoho odborníků v tématu internetové bezpečnosti si myslí, že je potřeba tyto informace co nejvíce zveřejňovat, a také podle toho konají. Existuje několik organizací, které se zasazují o publikaci těchto důležitých informací. Za tu nejzajímavější považuji společnost OWASP, která je nejaktivnější, co se týče publikování a podpory kvalitních a užitečných projektů.

Motivace útočníků můžeme dělit do tří skupin podle jejich přístupu k napadenému systému. První skupinu tvoří zájemci o zlepšení systému. Často to jsou bezpečnostní inženýři a penetrační testeři, kteří mají průniky do systému jako svou denní práci, díky které dostávají zaplaceno. Objednávají si je zejména bohatší a velké firmy, které si mohou zaplatit bezpečností audit specializovanou firmou.

Druhou skupinou jsou nadšenci, které baví pozorovat a rozebírat systémy okolo sebe. Tito lidé se webovým útokům a zranitelnostem věnují ve svém volném čase, z čiré radosti z poznání a překonání sebe sama.

Třetí skupina zahrnuje například takzvané crackery - lidi, kteří jsou už za hranicí práva a morálky. Jejich hlavním cílem je obohatit se na úkor druhých, často nezkušených uživatelů, kteří neumí či nechtějí zabezpečit svůj systém. Neváhají za tímto účelem překročit zákony a zneužít důvěřivosti lidí. Balancují na hranici práva a často jsou už za jeho hranou.

Zejména kvůli poslední skupině je třeba, aby informace o webových útocích byly snadno dostupné a co nejvíce šířené. Nejvíce zranitelní jsou totiž nevědoucí či nezkušení programátoři a administrátoři sítí, serverů a další informatické infrastruktury.

Software Safer je psaný jako jednoduchá aplikace, na které jsou dobře vidět jednotlivé principy fungování vnitřních funkčních součástí. Druh systému byl zvolen tak, aby umožňoval velké množství vektorů útoku, a byl obtížně zabezpečitelný. Pro správné zabezpečení programátor musí znát a poznat nejen všechny způsoby, musí je také správně ošetřit a zabezpečit takovým způsobem, aby se uživatelům se systémem pokud možno dobře pracovalo.

Safer je zajímavá aplikace z hlediska porovnání špatných a dobrých implementací. Je možné zde porovnávat téměř totožné třídy starající se o stejnou věc, kdy jedna je nezabezpečená a druhá zabezpečená, a studovat, v čem přesně se liší a co všechno je třeba neopomenout, aby třída byla implementována v sytému správně a opravdu danému webovému útoku zabraňovala.

Při stavbě bezpečnostní laboratoře jsme zjistili, že reálné zabezpečení trvá dlouho a vyžaduje velkou část znalostí a píle. Pro správnou implementaci všech zabezpe-

čovacích prvků se potřeba umět základy administrátorské práce a mít přehled o současných způsobech a trendech v oboru.

Stoprocentní ochrana jakéhokoliv systému či aplikace není možná – vše je výsledkem ochoty do zabezpečení investovat čas, znalosti a další prostředky. A jen tehdy, pokud jsou správně odhadnutá rizika, zabezpečení se chopí věci znalý odborník, má na něj dostatek času i financí a nezapomene při tom na koncového uživatele, potom můžeme mluvit o správném a bezpečném síťovém provozu.

## 8.1 Limity této práce

Tato práce samozřejmě neobsahuje všechny možné známé útoky - na to by její rozsah nestačil. Prezentuje pouze základní webové útoky, které se mohou vyskytnout v široké řadě systémů. Další rozšíření práce by mohlo zahrnout více druhů těchto útoků, popřípadě demonstraci jejich variability a nabídnutí více verzí jednoho mechanismu.

V systému Safer nejsou dobře vyřešeny interakce mezi jednotlivými nebezpečnými implementacemi a je možné, že mezi sebou budou nepředvídatelně interagovat. Konkrétní problém by pomohly odhalit automatizované testy, které by mohly být zajímavým rozšířením této práce. Pro garanci bezproblémového použití doporučuji zapnout vždy jen jednu zranitelnost najednou.

V práci jsem nezahrnul takzvané zero-day útoky, které spočívají v náročném a velmi podrobném zkoumání již existujícího softwaru a stavbě malwaru přímo pro nalezenou konkrétní zranitelnost, kterou zatím nikdo jiný nezná. Tato metoda totiž není pro účely této práce vhodná, avšak bylo by možné toto téma v budoucnu zpracovat.

V práci není obsažena také velká skupina útoků patřících do oboru sociálního inženýrství. Z praxe se jedná o nejúspěšnější druh útoků na systémy, vyžaduje však spíše dovednosti z oblasti soft skills a manipulace s obětí. Tato skupina útoků je vhodná pro statistické zkoumání na vzorku lidí, což je směr, kam by se mohla tato práce ubírat dále.

# Kapitola 9
# Závěr

V práci je zpracováno téma webových útoků. Představuje je k pochopení po technické stránce a vysvětluje jejich účel a způsob použití. Zjištěné skutečnosti se neváží pouze na zdravotnictví, ale jsou platné ve všech odvětvích silně spoléhajících na informace a webovou bezpečnost.

Dále k těmto druhům útoků zpracovávám jednotlivé konkrétní úlohy pro studenty předmětu Bezpečnost přenosu a zpracování dat, kteří budou mít tímto možnost si vybrané webové útoky sami vyzkoušet. Studenti si splněním úloh mohou dotknout reálného průběhu webového útoku a nastudovat správnou obranu proti němu.

Práce také představuje aplikaci Safer, která je ve verzi 1.0.1. Aplikace je plně funkční a dá se použít při nácviku webových útoků jako legální cíl. Na aplikaci se též dá studovat, jakým způsobem je implementována správná ochrana proti vybraným útokům. Je také snadné na aplikaci porovnat správný a špatný způsob použití ochranných prvků a srovnat, jaké detaily hrají roli ve správném zabezpečení.

Celá aplikace Safer včetně materiálů pro provádění webových útoků je uvolněna pod licencí MIT.

V práci také nalezneme popis stavby laboratoře. Ta slouží pro vytvoření prostředí pro bezpečné studování procesů webových útoků bez ohrožení cizích systémů či dat.

Budu rád, když se aplikace, úlohy i bezpečnostní laboratoř budou nadále rozvíjet, zejména když se budou přidávat nové druhy webových útoků a dále rozšiřovat zaměření laboratoře do větší šíře bezpečnostní problematiky.

# Literatura

Všechny odkazy jsou aktuální k datu podpisu autora.


[1] *R. Shirey, Internet Security Glossary, květen 2000.*
https://tools.ietf.org/html/rfc2828.

[2] *CNSS Instruction No. 4009 National Information Assurance Glossary, 26. 4. 2010.*
https://www.hsdl.org/?view&did=7447.

[3] *ENISA glossary, 2017.*
https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/glossaryG52.

[4] *Free On-Line Dictionary of Computing, 15. 02. 1995.*
http://foldoc.org/Application%20Program%20Interface.

[5] *PC Magazine, Encyclopedia, 28. 6 2009.*
http://www.pcmag.com/encyclopedia/term/37856/api.

[6] *Tomáš Klíma, web JakPsátPHP, článek Architektura MVC, 11. 4. 2017.*
http://jakpsatphp.cz/MVC/.

[7] *Eric Raymond, The Early Hackers, 2000.*
http://www.catb.org/~esr/writings/cathedral-bazaar/hacker-history/ar01s02.html.

[8] *Douglas Wilhelm, Professional Penetration Testing, kapitola 2: Ethics and Hacking, ISBN: 978-1-59749-425-0, 2010.*

[9] *Eric S. Raymond, Jargon File: Cracker, 29. 12. 2003.*
http://catb.org/jargon/html/C/cracker.html.

[10] *Web PC Tools firmy Symantec, What are crackers and hackers?, 2017.*
http://www.pctools.com/security-news/crackers-and-hackers/.

[11] *Zákon č. 101/2000 Sb., o ochraně osobních údajů, §4, článek a), 4. dubna 2000.*
https://www.uoou.cz/files/101_cz.pdf.

[12] *Zákon č. 101/2000 Sb., o ochraně osobních údajů, §4, článek b), 4. dubna 2000.*
https://www.uoou.cz/files/101_cz.pdf.

[13] *Webová stránka společnosti OWASP, 2017.*
https://www.owasp.org/.

[14] *Společnost OWASP na české wikipedii, 2017.*
https://cs.wikipedia.org/wiki/OWASP.

[15] *Seznam OWASP Top 10 roku 2013, verze Final Release, 12. 6. 2013.*
https://www.owasp.org/images/f/f8/OWASP_Top_10_-_2013.pdf.

[16] *Seznam OWASP Top 10 roku 2017, verze Release Candidate, 10. 5. 2017.*
https://github.com/OWASP/Top10/raw/master/2017/OWASP%20Top%2010%20-%202017%20RC1-English.pdf.

[17] *Webová stránka projektu OSSTMM, 2017.*
http://www.isecom.org/research/.

[18] *Webová stránka projektu Hacker Highschool, 2017.*
http://www.hackerhighschool.org/.

[19] *Webová stránka organizace PTES, 2017.*
http://www.pentest-standard.org/.

[20] *Martin Horyna, Krátký náhled na současnou problematiku kybernetických útoků, 29. 9. 2015.*
http://www.pravniprostor.cz/clanky/ostatni-pravo/kratky-nahled-na-soucasnou-problematiku-kybernetickych-utoku.

[21] *Stuxnet worm "targeted high value Iranian assets", 29. 9. 2010.*
http://www.bbc.com/news/technology-11388018.

[22] *Snowden and Greenwald: The Men Who Leaked the Secrets, RollingStone, 4. 12. 2013.*
https://www.rollingstone.com/politics/news/snowden-and-greenwald-the-men-who-leaked-the-secrets-20131204.

[23] *CVSS specifikace, CVSS v3.0 Equations, 2015.*
https://www.first.org/cvss/specification-document8-CVSS-v3-0-Equations.

[24] *CVSS specifikace, 2.3. Impact Metrics, 2015.*
https://www.first.org/cvss/specification-document2-3-Impact-Metrics.

[25] *firma Microsoft, TechNet Blog CZ/SK, 2017.*
https://blogs.technet.microsoft.com/technetczsk/p/microsoft-hyper-v/.