VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SOUBOR LABORATORNÍCH ÚLOH K DEMONSTRACI POČÍTAČOVÝCH ÚTOKŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

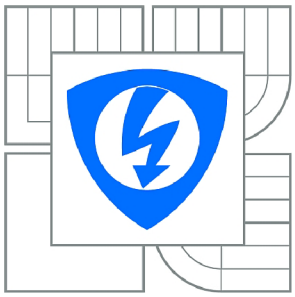
AUTOR PRÁCE
AUTHOR

Bc. MATOUŠ PLAŠIL

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

SOUBOR LABORATORNÍCH ÚLOH K DEMONSTRACI POČÍTAČOVÝCH ÚTOKŮ

COLLECTION OF LABORATORY WORKS FOR DEMONSTRATION OF COMPUTER ATTACKS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

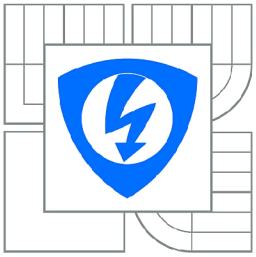
AUTOR PRÁCE — Bc. MATOUŠ PLAŠIL

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. KAREL BURDA, CSc.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Matouš Plašil ***ID:*** 134383
Ročník: 2 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Soubor laboratorních úloh k demonstraci počítačových útoků

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte a popište publikované útoky na počítače a počítačové sítě. U jednotlivých útoků vysvětlete jejich princip, uveďte potřebné podmínky a rekvizity a popište detailní postup při provádění útoku. Popsané útoky zhodnoťte z hlediska možnosti jejich praktické demonstrace pomocí virtuálních strojů na jediném počítači. Následně pro vybrané útoky vytvořte laboratorní úlohy, v nichž by si studenti mohli pomocí virtuálních strojů ověřit provádění těchto útoků. Pro laboratorní úlohy připravte software, videoukázky, návody pro studenty a dokumentaci pro vyučujícího.

DOPORUČENÁ LITERATURA:

[1] McClure, S.; Scambray, J.; Kurtz, G.: Hacking exposed. McGraw Hill. N. York 2012.
[2] Kim, P.: The Hacker Playbook: Practical Guide To Penetration Testing. CreateSpace Independent Publishing Platform. N. Charleston 2014.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 26.5.2015

Vedoucí práce: doc. Ing. Karel Burda, CSc.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Diplomová práce popisuje publikované útoky na počítače a počítačové sítě. Je vysvětlen princip metod pro shomažďování informací o cíli, jako je zjišťování dostupnosti, detekce OS, skenování portů. Další část popisuje útoky na důvěrnost, autentičnost a dostupnost. V praktické části jsou vytvořeny čtyři laboratorní úlohy a virtuální prostředí, pomocí kterého je možné prakticky otestovat útoky jako je ARP spoofing, DNS spoofing, SSL strip, Cross-site scripting, SQL injection, záplavové útoky (TCP, ICMP, UDP), TCP reset a útok na operační systém s využitím backdooru pomocí frameworku Metasploit. V rámci praktické části byly také vytvořeny videoukázky a dokumentace pro vyučující.

KLÍČOVÁ SLOVA

Počítačové útoky, Kali Linux, VMware Player, ARP spoofing, DNS spoofing, Cross-site scripting, SQL injection, ICMP flood, TCP, flood, UDP flood, TCP reset, framework Metasploit, PHP, MySQL, HTML, HTTP, HTTPS, CSS, Javascript

ABSTRACT

Diploma thesis describes published attacks on computers and computer networks. Principles of footprinting such as availability check, OS detection, port scanning were described. Next part explains attacks on confidentiality, integrity and availability. In the practical part were created four laboratory tasks and a virtual environment which allowed testing of ARP spoofing, DNS spoofing, SSL strip, Cross-site scripting, SQL injection, flooding attacks (TCP, ICMP, UDP), TCP reset and attack on operating system using backdoor with Metasploit framework. In practical part were also created video samples with attacks and documentation for teachers.

KEYWORDS

Computer attacks, Kali Linux, VMware Player, ARP spoofing, DNS spoofing, Cross-site scripting, SQL injection, ICMP flood, TCP, flood, UDP flood, TCP reset, framework Metasploit, PHP, MySQL, HTML, HTTP, HTTPS, CSS, Javascript

PLAŠIL, Matouš *Soubor laboratorních úloh k demonstraci počítačových útoků*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 166 s. Vedoucí práce byl doc. Ing. Karel Burda, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Soubor laboratorních úloh k demonstraci počítačových útoků“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského 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, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé diplomové práce doc. Ing. Karlu Burdovi, CSc. za odborné vedení, konzultace a podnětné návrhy k práci.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

S příchodem počítačů přišli i lidé snažící se do hloubky pochopit jejich fungování. Na základě znalostí vytvářeli programy označované jako viry. Zpočátku se jednalo o neškodné kusy kódu často určené pouze pro pobavení, netrvalo však dlouho a vznikly první viry způsobující škody. S příchodem internetu se tato problematika stala ještě rozsáhlejší, vznikali počítačoví červi.

Kybernetické útoky již nejsou doménou pouze nadšenců či frustrovaných zaměstnanců, je to obor profesionálu. Útoky jsou často vedené týmem, který útok přesně zacílí. V dnešní době je většina lidí neustále připojena k internetu pomocí chytrých telefonů, tabletů a dalších. Firmy a státní organizace využívají rozsáhlé počítačové systémy. Otázkou současnosti již není, zda k útoku dojde, ale kdy a za jakých okolností. Počítačová bezpečnost se stala seriózním problémem.

Diplomová práce v první části popisuje metody pro shromažďovaní informací o cíli, které jsou úzce svázány s počítačovými útoky. Popsány jsou způsoby zjišťování dostupnosti, detekce operačního systému, skenování portů a další mechanizmy sloužící k zacílení útoku.

V další části práce jsou popsány známé útoky na počítačové systémy. Popsány jsou útoky na důvěrnost, autentičnost a útoky na dostupnost. U jednotlivých útoků je vysvětlena nezbytná teorie pro pochopení útoku, princip a možná realizace pomocí současných aplikací. Zmíněny jsou jak dlouhodobě známé útoky, jako je Přetečení na zásobníku, tak i dnes relativně nové a využívané útoky typu Cross-site Scripting (XSS), SQL Injection.

V rámci praktické části proběhlo vytvoření virtuálního testovacího prostředí za pomoci programu VMware. Ve virtuálním prostředí jsou vybrané útoky otestovány. Z realizovaných útoků je následně vytvořen soubor čtyř laboratorních úloh pro výuku v rámci počítačových cvičení.

1 ZÁKLADNÍ POJMY

1.1 Počítačový útok

Počítačový útok je takový útok, při kterém se neoprávněná osoba pokouší získat, změnit či zničit majetek nebo data informačního systému. Obětí se zde stává majitel počítače (v nejčastějším případě firma), na který byl útok vykonán. Počítačový útok je nelegální, nemorální akt a je postihován zákonem dle §230 (Neoprávněný přístup k počítačovému systému a nosiči informací) [1]. Na začátek je důležité si vymezit dva základní typy počítačových útoků.

Prvním typem je pasivní útok. Je to takový útok, při kterém útočník nejčastěji odposlouchává komunikaci a získává informace o vybraném cíli. Útočník negeneruje žádná data k cíli. Tento typ útoku je složité odhalovat, jelikož při úspěšném útoku a při běžném používání cílového systému budou všechny funkce pracovat dle očekávání. Jedinou možností je využití IDS systému (Intrusion Detection System), což je systém, který monitoruje síťový provoz a snaží se odhalit podezřelé aktivity.

Naopak při aktivním útoku probíhá generování dat k cíli. Útočník se v tomto případě snaží buď o omezení dostupnosti cílového systému nebo se snaží odchytit a měnit nebo mazat data. Typickým útokem pro omezení dostupnosti může být útok DoS (Denial of service)

1.2 Penetrační testování

Penetrační testování je forma etického hackingu. Etickým hackingem je rozuměno hledání slabých míst v systému pro zjištění bezpečnostních rizik. Etický hacking nezneužívá zranitelností, v mnoha případech probíhá na požádání vlastníka systému. Vlastníkem může být například banka, která se chystá uvést informační systém do veřejného provozu, ale před oficiálním spuštěním si nechá provést penetrační testování pro zanalyzování bezpečnostních rizik.

Hlavním úkolem při testování je vniknutí do počítačové sítě či systému, odtud je dále snaha získat např. rootovský účet na důležitých unixových systémech, získání hesel z databází, firemní dokumentaci, tajné dokumenty či číselnou kombinaci k firemnímu sejfu. K vniknutí do systému se používají simulace možných útoků, ty jsou tvořeny na základě známých či neznámých slabin systému. Úspěšně provedené útoky napomáhají k posouzení rizika, které bezpečnostní chyby pro firmu představují.

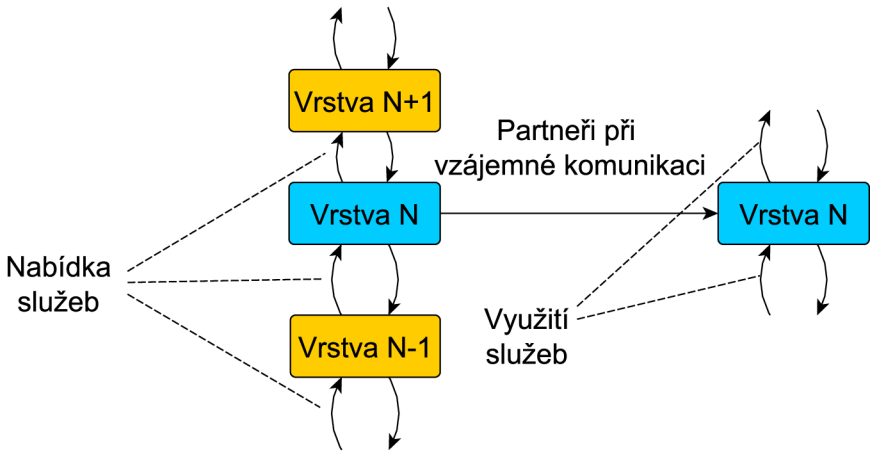
Jakmile je testování dokončeno, jsou získané údaje konzultovány s firmou, která si test zadala. Další důležitou částí hodnocení je navrhnutí způsobu, jakým informační systém ochránit před zjištěnou zranitelností, a tím i zvýšit bezpečnost celého

systému. Jelikož bezpečnost celého počítačového systému je taková, jak je zabezpečený jeho nejslabší článek[2].

1.3 Referenční model ISO/OSI

Referenční model ISO/OSI byl navrhnut v roce 1979 jako koncept pro vývoj standardů pro vzájemné propojování otevřených systémů v rámci organizace ISO. Základní myšlenkou ISO/OSI je dekompozice základního programového vybavení sítě na hierarchicky uspořádané vrstvy. Jak je vidět na obrázku 1.2, je těchto vrstev v ISO/OSI definováno celkem sedm. Každá vrstva N má na starost přesně určený okruh úkolů. Mechanismy, které jsou na jedné vrstvě, se pak nabízí jako služby vrstvě N+1, tedy vyšší vrstvě. K realizaci těchto služeb využívá i služeb nižších vrstev N-1 viz obr. 1.1.

Stejnolehlé vrstvy mají definována přesná pravidla při vzájemné komunikaci. Musí spolu být domluveny na společných pravidlech při komunikaci a tyto pravidla musí důsledně dodržovat. Takový soubor pravidel, který stejnolehlé vrstvy ISO/OSI modelu používají, tvoří protokol. Pro jednu vrstvu může být definované větší množství protokolů, pro úspěšnou komunikaci stejnolehlých vrstev však musí obě vrstvy komunikovat pomocí stejného protokolu. V momentě, kdy je pro každou vrstvu definován jeden konkrétní protokol, vzniká soustava protokolů (protocol stack) [4].



Obr. 1.1: Schéma vzájemné komunikace vrstev

1. **Fyzická vrstva (Physical Layer)**

   Zajišťuje přenos jednotlivých bitů mezi příjemcem a odesílatelem pomocí fyzické cesty. Je potřeba vyřešit otázky převážně technického charakteru, jako je například úroveň napětí, pomocí kterého bude reprezentována logická jednička, nula, případně jak dlouho bude trvat jeden bit. Na této vrstvě se také řeší například typy kabeláže, konektory kabelů, typy signálů, pomocí kterých budou data přenášena apod. Na této vrstvě se přenášejí bity.

2. **Linková vrstva (Data Link Layer)**

   Linková vrstva (často nazývaná také jako spojová) poskytuje za pomoci služeb fyzické vrstvy bezchybný přenos celých bloků dat. Má za úkol rozpoznat začátky a konce jednotlivých rámců, kontroluje, zda byly rámce přenesené správně pomocí kontrolních součtů, jako je například CRC. Provádí potvrzování bezchybně přijatých rámců a v případě chybně přijatých žádá o jejich opětovné vysílání. Na této vrstvě se tedy přenášejí rámce.

3. **Síťová vrstva (Network Layer)**

   Hlavním úkolem síťové vrstvy je směrování. Jelikož linková vrstva zajišťuje pouze přenos celých rámců mezi dvěma uzly, je potřeba mechanismus pro nalezení cesty k cíli. Síťová vrstva tedy zajišťuje volbu vhodné trasy (route) přes mezilehlé uzly a také postupné předávání jednotlivých paketů od odesílatele až ke konečnému příjemci. Na této vrstvě se přenášejí pakety.

4. **Transportní vrstva (Transport Layer)**

   Transportní vrstva vytváří iluzi přímého spojení (odstiňuje tedy skutečnou topologii). V případě příjmu paketů se stará o sestavování na původní data, v opačném případě rozděluje odesílané data na jednotlivé pakety. Díky tomuto mechanismu je tedy možné zajistit přenos libovolně velkých zpráv, přestože mají pakety omezenou velikost.

5. **Relační vrstva (Session Layer)**

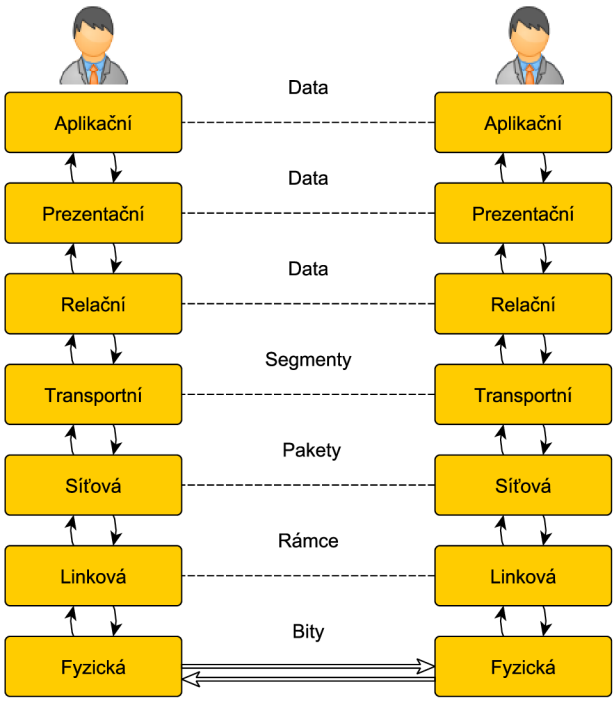
   Úkolem této vrstvy je navazování, udržování a rušení relací mezi koncovými účastníky. Při navazování relace využívá tato vrstva služeb transportní vrstvy, u které žádá o vytvoření spojení. Prostřednictvím vytvořeného spojení poté probíhá komunikace mezi oběma účastníky relace. V případě nutného řízení určuje, kdo má kdy vysílat.

6. **Prezentační vrstva (Presentation Layer)**

   Provádí potřebné konverze přenášených dat, jako je například převod ASCII na EBCDIC. Případně provádí kompresi přenášených dat nebo jejich šifrování.

7. **Aplikační vrstva (Application Layer)**

   Aplikační vrstva umožňuje aplikacím přímý přístup ke komunikačnímu médiu. Na této vrstvě fungují služby a protokoly jako například: FTP, SSH, DNS, IMAP apod.



Obr. 1.2: Komunikace mezi dvěma uživateli pomocí modelu ISO/OSI

1.4 IP adresa

IP adresa slouží k adresování síťového prostoru a spadá do síťové vrstvy TCP/IP modelu. IP adresa je pouze abstraktní adresou, která musí být posléze převedena na skutečnou fyzickou adresu MAC, nejčastěji za pomoci protokolu ARP. IP adresa je dvousložková, tvořená adresou dílčí sítě a číslem daného hostitelského počítače. Velikost těchto dvou částí udává maska sítě. IP adresy verze 4 jsou 32-bitové, dají se vyjadřovat binárně, ale z hlediska srozumitelnosti pro člověka je zavedená konvence v podobě decimálního zápisu a tečkovou desítkovou notací.

Tato notace spočívá v tom, že se 32 bitů IP adresy rozdělí na 4 části, tímto vzniknou „oktety“. Každá část je pak vyjádřená jako celé desítkové číslo v rozsahu 0 až 255 [3].

Maska sítě

Maska sítě určuje bity vyhrazené pro adresu sítě. Má stejný zápis jako IP adresa. Binárním násobením IP adresy hosta a masky je možné zjistit IP adresu dané sítě.

Například maska sítě 255.255.0.0 má v prvních dvou oktetech pouze jedničky a v posledních dvou pouze nuly. V případě, že by host měl IP adresu 147.229.151.1

a masku výše uvedenou, tak bychom po binárním roznásobení získali adresu 147.229.0.0, což je adresa sítě.

Třídy IP adres

IP adresy se dělí na třídy. Podle původní filozofie TCP/IP počítá se třídami (C),(B) a (A), viz tabulka 1.1. Toto řešení se později ukázalo jako neefektivní, efektivnějším se ukázalo řešení pomocí podsíťování [3].

Tab. 1.1: Rozdělení tříd IP adres [3]

Třída	Rozsah prvního oktetu adresy	Dělení adresy na adresu Sítě a Hosta	Standardní maska sítě	Počet možných sítí / hostů na jednu síť
A	0 - 127	S.H.H.H	255.0.0.0	128 / 16 777 214
B	128 - 191	S.S.H.H	255.255.0.0	16 383 / 65 534
C	192 - 223	S.S.S.H	255.255.255.0	2 097 150 / 254
D	224 - 239		Multicastové adresy	
E	240 - 255		Experimentální adresy	

Adresa sítě

Jak již bylo řečeno výše, adresa sítě vznikne logickým součinem IP adresy hosta a masky sítě.

Všesměrová adresa (broadcast)

Jedná se o IP adresu pomocí které lze odeslat paket všem stanicím ležícím v dané síti. Vznikne tak, že všechny bity tvořící adresu hosta v rámci IP adresy se nahradí binární 1.

Lokální smyčka (loopback)

Jedná se softwarovou smyčku uvnitř počítače. Má IP adresu 127.0.0.1. Pakety s touto adresou nikdy neopustí počítač - vhodné např. pro meziprocesorovou komunikaci.

1.5 MAC adresa

MAC adresa, někdy nazývaná také jako fyzická adresa, slouží jako jednoznačný identifikátor síťového zařízení. Ačkoli by měla být MAC adresa unikátní, lze u dnešních síťových karet MAC adresu změnit.

MAC adresa má délku 48 bitů, skládá se ze čtyř čísel zapsaných v hexadecimálním tvaru, které jsou odděleny buď pomlčkami, nebo dvojtečkami. Příklad MAC adresy:

```
F4-09-D8-A4-90-50
```

Adresa se dělí na dvě části, první tři oktety značí OUI (Organizationaly unique identifier), což je unikátní identifikátor výrobce. Každý výrobce síťového zařízení

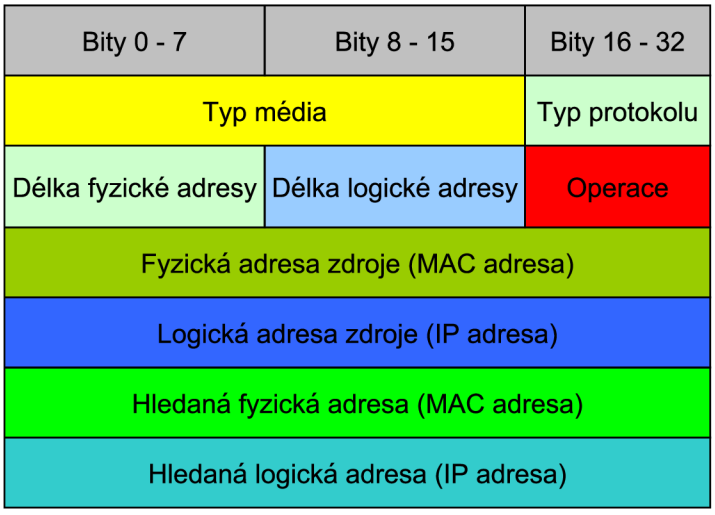
má svůj unikátní OUI, který používá a podle kterého je zpětně vyhledatelný. OUI je možné získat od organizace IEEE. List s OUI a výrobci je dostupný na následující adrese `http://www.ieee.org/netstorage/standards/oui.txt`. Z toho vyplývá, že výše uvedená MAC začínající na `F4-09-D8` náleží společnosti Samsung. Druhé tři oktety jednoznačně identifikují síťovou kartu.

1.6 ARP protokol

Protokol ARP transformuje adresy vyšší úrovně (IP adresy) na adresy nižší úrovně (MAC adresy). Úkolem ARP protokolu je tedy nalézt odpovídající fyzickou adresu na základě IP adresy, která je nezbytná pro úspěšnou komunikaci.

ARP používá pomocné tabulky dočasných záznamů (uložených v cache). Do těchto tabulek ukládá PC nebo síťový prvek již známé IP adresy a jejich fyzické adresy. Tyto záznamy se naplní buď pomocí ARP protokolu (pokud potřebuje stanice získat fyzickou adresu jiné stanice za účelem odeslání paketu) nebo se dají zadat tyto informace manuálně [3].

U operačního systému Windows lze ARP cache zobrazit pomocí příkazu: **`arp -a`**. Vložený údaj lze z tabulky smazat pomocí příkazu **`arp -d X.X.X.X`**, kde X.X.X.X je IP adresa, pro kterou bude záznam vymazán. Pro vložení statického záznamu zle využít příkazu **`arp -s X.X.X.X Y-Y-Y-Y-Y-Y`**, kde X.X.X.X značí IP adresu a Y-Y-Y-Y-Y-Y fyzickou adresu. ARP paket má následující strukturu:



Obr. 1.3: Struktura ARP paketu [3]

Typ média
: Hodnota indikující použité médium (např. Ethernet má hodnotu 0x0001, ATM 0x0010).

Typ protokolu (Protocol type)
: Určení typu vyššího protokolu, v rámci kterého se logická adresa používá (např. IP má hodnotu 0x8000).

Délka fyzické adresy (Hardware length)
: Určení délky fyzické adresy v bajtech (např. Ethernet 0x06).

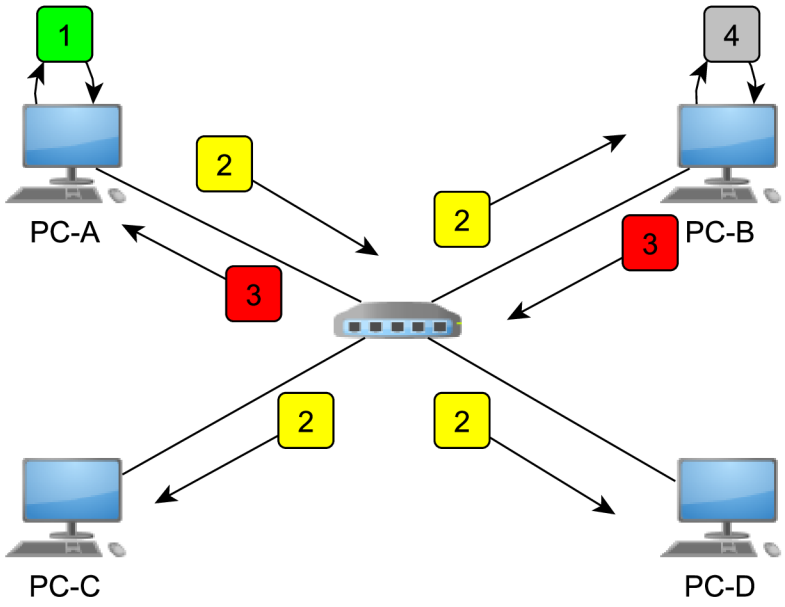
Délka logické adresy (Protocol length)
: Určení délky logické adresy v bajtech (např. IPv4 0x04).

Operace (Operation)
: Specifikace operace, kterou odesílatel paketu provedl (např. hodnota 0x0001 značí požadavek, hodnota 0x0002 odpověď).

Fyzická adresa zdroje / hledaná (Sender / target hardware address)
: Obsahuje fyzickou adresu zdroje / hledanou.

Logická adresa zdroje / hledaná (Sender / target logical address)
: Obsahuje logickou adresu zdroje / hledanou.

Na níže níže uvedeném obrázku 1.4 je znázorněn proces komunikace při ARP požadavku.



Obr. 1.4: Schéma komunikace (ARP request)

1. PC-A chce odeslat data PC-B. V prvním kroku si PC-A prozkoumá svojí tabulku (ARP cache), zda nenajde informaci odpovídající hledané síťové adrese.
2. V případě, že PC-A nenalezne záznam o adrese PC-B, tak vyšle žádost (request) ve formě broadcastu do sítě. Tuto žádost obdrží všechny PC v rámci broadcastové domény.
3. Až na PC-B všechny stanice žádost zahodí - PC-B odešle odpověď (response) ve formě unicastu PC-A s vyplněným polem hledané fyzické adresy.
4. PC-B si zároveň zkontroluje obsah své tabulky a případně jí doplní o informaci obdrženou v žádosti ARP od PC-A.

2 SHROMAŽĎOVÁNÍ INFORMACÍ O CÍLI

Shromažďování informací je jedním ze základních kroků útočníka před provedením útoku. Při systematickém a metodickém získávání informací o cíli dochází k vytvoření ucelené představy o zabezpečení cílového systému. Tyto informace o cíli jsou často získané bez nutnosti komunikace s cílovým serverem, jelikož mnoho informací lze získat od třetích stran. Útočníci mají velkou motivaci v získání velkého množství těchto informací. Navíc je v současné době možné získat mnoho těchto informací legální cestou.

2.1 Veřejně dostupné informace

Při zjišťování veřejně dostupných informací se využívá různých metod a technik, které se liší v závislosti na prostředí, ve kterém se cíl nachází. Těmito prostředími jsou internet, intranet, vzdálený přístup či extranet.

Webové stránky

Nejvhodnějším začátkem pro získání informací o cíli jsou webové stránky. Jelikož v mnoha případech organizace umísťují na své webové stránky více informací, než je ve skutečnosti nutné. Prohlížení HTML kódu se jeví jako zajímavý odrazový můstek, obzvláště pokud vývojář zanechal v kódu komentáře.

Protože není efektivní procházet stránku po stránce ve webovém prohlížeči, tak existují nástroje pro website mirroring. Jedná se o nástroje, které se snaží stáhnout kompletní webovou stránku včetně souborů, obrázků a skriptů. Mezi nejznámější dvě aplikace patří **Wget** (platforma Linux) a **TeleportPro** (platforma Windows). Jedná se o jednoduché nástroje, kterým stačí zadat pouze URL adresu cílové stránky a spustit stahování.

V některých případech není jednoduché stáhnout kompletní webovou stránku výše uvedeným způsobem, například v případě skrytých souborů s skriptů, na které není odkazováno. V tomto případě existuje nástroj **DirBuster**, který využívá techniku rekurzivního vyhledávání hrubou silou. Tato funkce z něho činí sice nástroj efektivnější, ale zároveň nápadnější, jelikož používá hrubou sílu. DirBuster umožňuje nastavení proxy, poskytuje tedy možnost krytí útočníka [5].

V případě organizace je zajímavou volbou kromě testování oficiálních webových stránek vyzkoušet i jiné často používané www adresy. Těmi jsou `www1, web, web1, test, test1`. Tyto adresy jsou hojně používané. Další zajímavou adresou je:

`https://owa.adresa.com či https://outlook.adresa.com.`

jedná se o proxy pro přístup do intranetu pro Microsoft exchange servery. V neposlední řadě lze zkusit také VPN spojení, které využívá nejčastěji adresy:

`http://vpn.adresa.com, https://vpn.adresa.com, http://adresa.com/vpn`

Na těchto stránkách lze obvykle najít informace o verzi VPN, výrobci, kontakt či možnost stažení klienta.

Příbuzné organizace

Mnoho organizací v dnešní době využívá outsourcing. Outsourcing znamená, že určitou část své práce předají jiné firmě. Hojně využívané je outsourcování tvorby webových stránek. Důsledkem toho je možné, že při nahlédnutí do zdrojového kódu HTML či CSS lze například odhalit organizaci, která tyto stránky vytvořila. Na základě těchto informací se příbuzná firma může stát zajímavým cílem pro útok [5].

Informace o adrese a zaměstnancích

Informace o zaměstnancích lze využít při sociálním inženýrství. V dnešní době sociálních sítí nelze informace o zaměstnancích podceňovat. Množství informací, které lze najít skrze Facebook, LinkedIn a podobné služby je velmi rozsáhlé.

V případě znalosti adresy se dá zjistit pomocí `maps.google.com` a **street view** detailní prostředí areálu firmy. Jelikož auta Google street view nesnímají pouze obraz, ale i MAC adresy přístupových bodů Wi-fi, je tedy možné na základě adresy získat informace o bezdrátové infrastruktuře. Tyto informace jsou dostupné na **Google Locations**.

V krajním případě lze získat informace z vyhozeného odpadu (popelnice, kontejnery). Pokud firma neskartuje dokumenty, tak je i zde možné riziko úniku dat.

Vyhledávací nástroje

Dnešní vyhledávací nástroje poskytují kvalitní výsledky, spousta moderních vyhledávačů poskytuje pokročilé vyhledávání (bing.com, yahoo.com, google.com). Pomocí vyhledávačů lze získat velké množství citlivých informací. Například vyhledávání gooogle: `allinurl: tsweb/default.html` odhalí weby, na kterých je spuštěn Microsoft Remote Desktop Web Connection. K nalezení jsou stovky takovýchto serverů.

Webová stránka `www.hackersforcharity.org/ghdb` (Google hacking Database) poskytuje databázi zranitelností, které se dají pomocí Google vyhledat. Webová stránka je staršího data, ale stále lze najít mnoho webů s těmito zranitelnostmi [5].

Zajímavým nástrojem je **SiteDigger** od firmy McAfee, který se dokáže zaměřit na určenou doménu a za použití známých zranitelností testuje web na jejich existenci [5].

2.2 Zjištění dostupnosti

Použitím programu ping se zjišťuje, zda je cílový host dostupný. Útočník odešle žádost **ping echo request** na cílového hosta a očekává jeho odpověď **echo reply**. Tím, že útočník obdrží **echo reply**, ví útočník, že je host dostupný. Ověřit dostupnost hosta lze pomocí protokolu ICMP, ale také pomocí protokolů ARP, TCP a UDP [5].

ARP host discovery

Tato metoda je vhodná pro použití v lokálních sítích LAN, využívá se zde vytvoření ARP dotazu.

1. ARP-SCAN
Je jednoduchá utilita, která zjistí přítomnost hostů v rámci určené podsítě. Zjistí všechny IP adresy, které odpovídají na (ARP dotaz) a jejich MAC adresy. Dále je schopná zjistit výrobce síťového rozhraní, které odpovídá z OUI (Organizationally unique identifie), pokud je k dispozici.

2. Network Mapper (nmap)
Jedná se o komplexní nástroj pro skenování sítě. Podporuje právě také i ARP ping, pokud se spustí s příznakem `-PR`. Stejně jako **ARP-SCAN** skenuje IP adresy v celé zadané podsíti.

3. CAIN
Aplikace pro platformu Windows. Kromě ARP host discovery umí také přepnout síťové rozhraní do odchytávajícího módu (sniffing) a odchytávat provoz pro zadanou MAC adresu.

ICMP host discovery

Klasickým způsobem, jak ověřit dostupnost hosta v prostředí internetu, je skrze protokol ICMP (internet control message protocol). Je to protokol navržený pro tyto účely a obsahuje hned několik typů zpráv pro diagnostiku a zjištění statusu hosta v prostředí internet.

Následující tabulka 2.1 zobrazuje používané typy zpráv, které jsou v rámci ICMP protokolu.

Tab. 2.1: Typy ICMP zpráv

Typ zprávy	Popis
0	Odpověd na žádost (echo reply)
3	Host je nedostupný
5	Přesměrování
8	Žádost na echo (echo request)
11	Vypršení času na odpověď
14	Žádost na zjištění času (Timestamp request)
15	Odpověď s časem (Timestamp reply)
17	Žádost o masku sítě
18	Odpověď na masku sítě

Nejčastějším způsobem pro zjištění dostupnosti je typ zprávy **8 Echo Request**, kterou generuje útočník a získává zprávu typu **0 Echo reply**. Mezi další užitečné typy zpráv patří **13 timestamp**, pomocí níž lze zjistit systémový čas hosta, a zpráva typu **17**, pomocí kterého se dá zjistit síťová maska podsítě, ve které host leží.

Pro realizaci ICMP host discovery lze využít utilitu **nmap**, kdy se za použití parametru `-SN` (no port scan) provede ICMP Echo request, ARP ping a ICMP Timestamp.

Zde je důležité uvést, že pokud cílový host disponuje systémem pro detekci vniknutí (IDS – intrusion detection system), tak tento krok může způsobit jeho spuštění. Nenápadnější cestou je v případě použití utility NMAP použití příznaku `-PE`, kdy se použije pouze klasický ICMP typ 8 (echo request).

Dalším zajímavým nástrojem je **nping**. Je to extrémně robustní nástroj pro host discovery a skenování portů. Jeho velkou výhodou je možnost podvržení zdrojové MAC adresy a cílové IP adresy při generování dotazů, což může útočník využít při maskování.

Dále je možné využít aplikaci **SuperScan** (platforma Windows) [5].

TCP hosty discovery

Jelikož v reálném prostředí je protokol ICMP z důvodu bezpečnosti zakázaný, lze využít protokolu TCP. Většina serverů poskytuje alespoň jednu spojovou službu na síti, jako je například webový server. Z toho důvodu se předpokládá, že je alespoň jeden port vždy otevřen a hostům je umožněno se připojit. Na serveru je tedy i firewall nakonfigurovaný tak, aby propustil tuto komunikaci. Touto cestou je tedy také možné ověřit dostupnost. Pro realizaci je možné využít např. utilitu **nmap**.

Lze využít výchozí port list (což je cca 1000 portů) a ty nechat proskenovat, nejedná se ale o nejelegantnější řešení už z důvodu nápadnosti při obrovském množství pokusů. Zde se jeví jako nejlepší možnost odhadnutí služeb a otestování nejčastěji využívaných portů pro tyto služby. Příklad často využívaných portů: **SMTP (25), POP(110), AUTH(113), IMAP(143)** [5].

2.3 Detekce operačního systému

Jednou z důležitých informací k realizaci útoku na operační systém je získání informací o jeho typu, v lepším případě i jeho verzi. Na základě těchto informací, lze zjistit, jaké slabiny daná verze OS skrývá, a ty poté zneužít. Existují dva základní způsoby detekce cílového OS a těmi jsou aktivní a pasivní detekce.

Aktivní detekce OS

První z nejjednodušších technik bylo využití uvítacích zpráv (banner-grabing).
Po úspěšném připojení na server (např. pomocí FTP, telnet, http apod.) byla odeslána klientovi uvítací zpráva, z které bylo možné získat informace o běžícím operačním systému.

Dalším způsobem může být odhadování operačního sytému na základě otevřených portů. Pokud má například cílový systém otevřený port 445, 139 a 135, jedná se s největší pravděpodobností o systém Windows, jelikož většina operačních systémů Windows má tyto porty otevřené. Některé porty jsou také specifické přímo pro operační systém, názornou ukázkou může být port 3389 (RDP) Remote Desktop Protocol, protokol pro připojení ke vzdálené ploše (není však pravidlem, že všechny operační systémy mají přístup ke vzdálené ploše povolený). Naopak UNIX systémy mají spuštěné služby jako například 111 (portmapper), port 512-514 (Berkeley services), 2049 (NFS) a podobné. Na základě prostého skenování otevřených portů lze tedy odhadnout typ použitého operačního systému.

Třetí způsob je poněkud komplexnější. Využívá odlišností (technika zvaná stack fingerprinting), které mají jednotlivé operační systémy u implementace TCP/IP modelu. Na základě znalosti těchto odlišností je vytvořen odhad operačního systému (neboli educated guess). Útočník tedy odešle paket a čeká na odpověď OS. Využívá se několik technik ke zjištění. Níže jsou pro příklad uvedeny čtyři metody:

Prozkoumání FIN - Odeslání paketu s příznakem FIN na otevřený port serveru. Dle specifikace RFC 793 by neměl server reagovat. Operační systémy Windows (Windows 7, XP,200X,Vista) však reagují odesláním FIN/ACK.

Zaslání nedefinovaného TCP - Cílem je odeslání paketu SYN s nedefinovaným příznakem v hlavičce TCP. Některé operační systémy, jako je např. Linux odešlou odpověď se stejným příznakem.

Velikost počátečního okna u TCP – Je sledována hodnota velikosti u počátečního okna. Velikost generuje operační systém na serveru a ve většině případů je hodnota pro každý typ OS unikátní.

Hodnota ACK - Je zkoumána hodnota ACK v přijatém paketu. Některé OS navracejí v ACK předešlou hodnotu sekvenčního čísla, jiné tuto hodnotu inkrementují o 1.

K praktické realizaci lze využít utilitu **Nmap** s příznakem `-O` , který slouží pro detekci cílového OS. Cílový příkaz ke zjištění běžícího OS na adrese **192.168.1.20** by mohl být následující:

```
nmap -O 192.168.1.20
```

Pomocí nástroje nmap je možné zjistit i typ OS u serveru, který nemá otevřený žádný port, ovšem s nižší přesností.

Nevýhodou aktivního způsobu je fakt, že útočník odesílá pakety přímo serveru a pokud server disponuje systémem IDS (systém detekce vniknutí), tak je útočník odhalen.

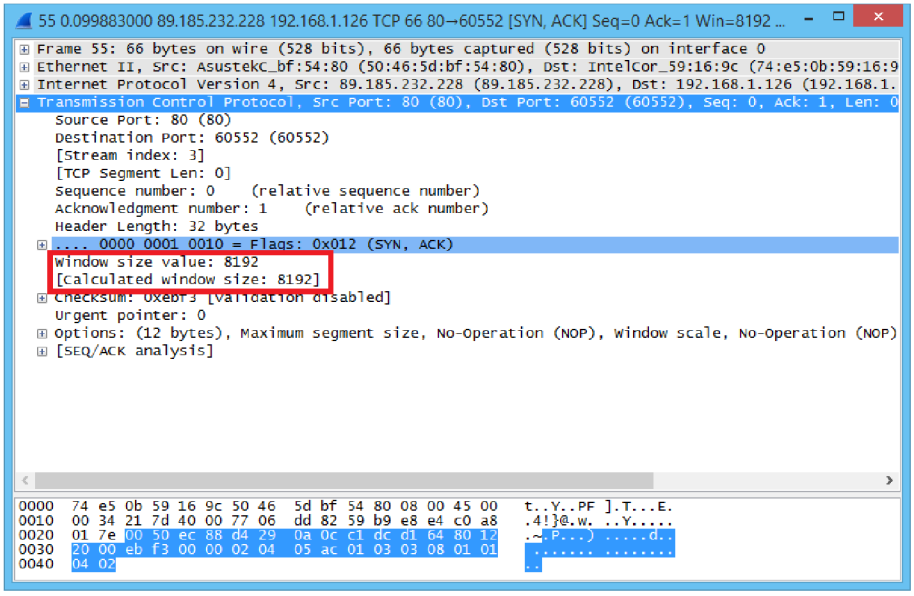
Pasivní detekce OS

Anonymnějším přístupem je pasivní detekce. Útočník odposlouchává komunikaci mezi serverem a komunikujícím klientem (například pomocí programu Wireshark). Tuto komunikaci poté podrobí analýze a na základě informací o velikosti počátečního okna u TCP protokolu, hodnoty TTL, případně hodnoty nastavené fragmentace určuje typ operačního systému. Nevýhodou této techniky je nutnost správného umístění útočníka v síti tak, aby byl schopný odposlechnout výše uvedenou komunikaci. V níže uvedené tabulce 2.2 jsou uvedeny známé hodnoty určujících parametrů u současně používaných operačních systémů.

Tab. 2.2: hodnoty parametrů generovaných OS

Operační systém (OS)	**Počáteční TTL (IP)**	**Velikost okna (TCP)**
Linux (kernel 2.4 a 2.6)	64	5840
Googlem upravený Linux	64	5720
FreeBSD	64	65535
Windows XP	128	65535
Windows 7, Vista a Server 2008	128	8192
Cisco Směrovač (IOS 12.4)	255	4128

Praktická ukázka odchycení komunikace: Klient s IP adresou **192.168.1.126** komunikuje s cílovým serverem **89.185.232.228**. Útočník zachytí níže uvedenou komunikaci pomocí programu **Wireshark**. Vyhledá paket odeslaný ze strany serveru s počáteční velikostí okna, viz obr. 2.1.



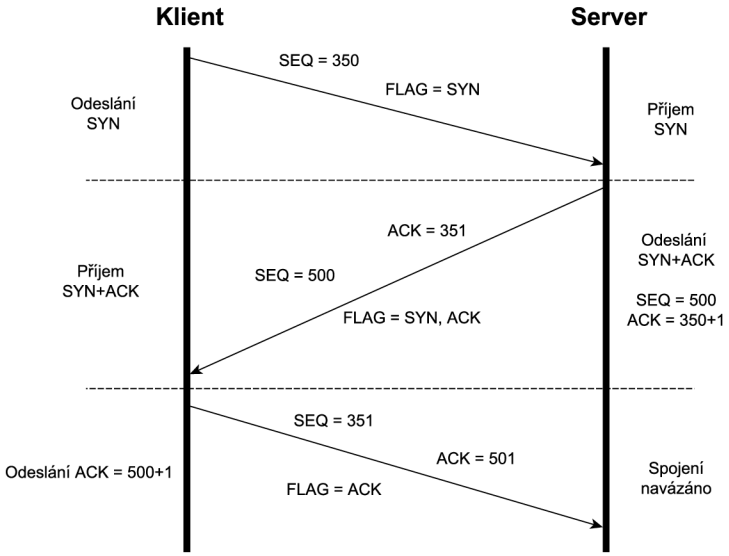
Obr. 2.1: Potvrzující paket (SYN,ACK) s velikostí okna

Na výše uvedeném obrázku je počáteční velikost okna **8192**, z čeho lze na základě uvedené tabulky 2.2 usuzovat, že jde o operační sytém Windows.

2.4 Skenování portů

Skenování portů je proces, při kterém jsou odesílány pakety protokolu TCP či UDP na cílový server, za účelem zjištění spuštěných služeb. Zjišťuje se, zda tyto služby na portech „naslouchají“. Identifikace portů je klíčová pro zjištění zranitelností systému. V případě znalosti cílového systému je snadné dohledat jeho verzi a slabá místa u běžících služeb. Pokud má systém bezpečnostní chybu, je jen otázkou času, kdy se na základě výše získaných informací útočník pokusí proniknout do systému.

Nejprve je nutná základní teorie TCP protokolu. TCP protokol (Transmision control protocol) je spojovanou službou a před odesláním dat vytváří spojení. Toto spojení je plně duplexní a přenášené pakety jsou číslovány. Pokud jsou data poškozená nebo ztracená, jsou znovu vyžádána. Jedná se tedy o spolehlivý přenos dat. Při samotném sestavování spojení se využívá trojcestná komunikace dle následujícího schématu 2.2:



Obr. 2.2: Schéma navazování spojení TCP [6]

V prvním kroku klient vytvoří inicializační paket SYN a ten jej společně s vygenerovaným sekvenčním číslem SEQ (např. 350) odešle na server. Nastaví si časovač pro opětovné odeslání a čeká na odpověď serveru.

Poté, co dorazí SYN paket na server, tak IP proces zjišťuje, zda je možné ho přijmout. V případě, že je vše v pořádku, proběhne kontrolní součet přijatého paketu. Server na základě stavu a verze operačního systému vygeneruje vlastní sekvenční číslo SEQ (např. 500) a přijaté ACK inkrementuje o 1, odesílá tedy ACK= 351.

V posledním kroku klient obdrží paket o potvrzení synchronizace (SYN+ACK) od serveru a posílá zpět potvrzení o přijetí. Tento odesílaný paket má nastavený příznak ACK (501) a SEQ (351). V momentě kdy server potvrzení přijme, je spojení sestaveno a může být zahájen přenos dat.

Způsobů, jak skenovat porty, existuje několik, většinou se liší mírou anonymity nebo druhem protokolu (TCP, UDP) [6].

TCP Connect skenování

Jedná se o klasické vytvoření TCP spojení. K realizaci tohoto skenování stačí program telnet, zadání IP adresy a cílového portu serveru. V případě, že je port otevřen, dojde k navázání spojení nebo naopak. Nevýhodou je, že toto skenování je velice nápadné. I když server, který je skenován nebude mít žádný program na detekci skenování portů (IDS), bude po navázání spojení uložena IP adresa v logovacích souborech. Servery ve většině případů zaznamenávají úspěšná spojení [7].

Pro tento typ skenování lze využít i program **nmap**, který je součástí Linuxové distribuce Kali Linux. Příkaz pro tento druh skenování vypadá následovně:

```
nmap -sT -p 0-65536 (ROZSAH_PORTU) IP_ADRESA_OBETI
```

TCP Stealth SYN skenování

Stealth SYN je neúplné Connect skenování. Aby bylo zjištěno, zda je port otevřen, není potřeba úplného trojcestného navázání spojení. Stačí, když klient odešle SYN a server odpoví SYN+ACK. Klient poté nepotvrdí navázání spojení příznakem ACK, spojení tedy není navázáno. Jelikož ale server odeslal SYN+ACK, lze považovat port za otevřený [7].

V dřívější době byl tento typ skenování neodhalitelný, jelikož neexistovali programy, které by byly schopné tuto aktivitu detekovat. V současné době je většina programů těchto druhů „aktivity“ v síti schopná detekovat, ovšem rozšíření těchto programů není příliš velké. Tento typ skenování lze opět provést pomocí programu **nmap** s parametry: [1]

```
nmap -sS -p 0-65536 (ROZSAH_PORTU) IP_ADRESA_OBETI
```

TCP FIN, X-mas a Null skenování

Principem těchto tří způsobů skenování je poslat neočekávaná data na porty. Poznat, zda je port otevřený, je u TCP jednoduché. V rámci protokolu TCP musí být před samotným odesláním sestaveno spojení, musí být správně nastavována a potvrzována sekvenční čísla. Tyto typy skenování buď generují náhodné čísla, nebo jsou nastaveny na nulu. Pokud na portu běží nějaká služba, příchozí pakety těchto vlastností bude server ignorovat. Pokud bude port uzavřen, server odešle zpět klientovi paket RST a ACK.

Bohužel takovéto skenování je na OS Windows XP SP2 a s největší pravděpodobností i na novější verze neúčinné. Takovýto neočekávaný paket odešlou zpátky, i když je port otevřen nebo zavřen.

Skenování X-mas posílá paket s příznaky FIN, URG a PSH. Název X-mas (Christmas) vznikl, protože má paket „rozsvícené“ velké množství příznaků a připomíná vánoční stromeček.

U skenování FIN se odesílá TCP paket pouze s příznakem FIN. V případě Null skenování se odešle TCP paket bez příznaku.

Ani tyto tři typy skenování nejsou v současné době příliš tajné a existují sledovací programy, které tyto druhy skenování odhalí [7].

[1] pro tento typ skenování je potřeba nmap spustit v linuxu s právy root.

Skenování lze provést pomocí programu **nmap**, kdy se pro X-mas skenování použije příkaz:

```
nmap -sX -p 0-65536 (ROZSAH_PORTU) IP_ADRESA_OBETI
```

Pro FIN skenování lze využít příznak `-sF`, pro Null `-sN`. Jak již bylo výše uvedeno, pokud je port otevřen, tak server ignoruje příchozí paket. Proto je vhodné odeslat paket vícekrát, aby se předešlo mylné informaci v případě, že by se paket ztratil.

TCP ACK skenování

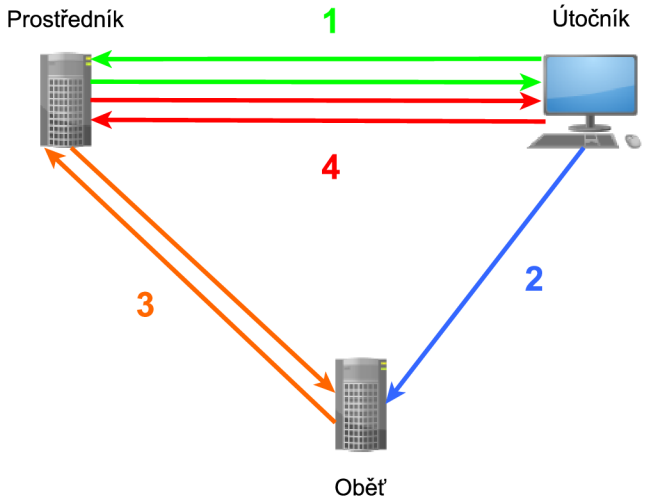
Tento typ skenování posílá TCP paket s příznakem ACK, sekvenční a příznak ACK jsou náhodně vygenerovány. Paket tedy není očekáván a server odešle zpátky paket s příznakem RST. Pokud je cílový port filtrován firewallem, je příchozí paket od klienta (útočníka) zachycen a server neodešle nic. Tímto skenováním není možné zjistit, zda je port otevřený, lze však zjistit, zda je port filtrovaný nebo ne [7].

```
nmap -sA -p 0-65536 (ROZSAH_PORTU) IP_ADRESA_OBETI
```

TCP Idle skenování

Jedná se o nejsložitější techniku skenování. Dovoluje však útočníkovi zcela anonymně skenovat oběť, aniž by se k oběti dostala útočníkova IP adresa. Útok využívá prostředníka, skrze kterého zjišťuje informace o stavu portů oběti.

Je důležité si uvědomit, že každý IP paket obsahuje identifikační číslo. Toto číslo je v rozmezí 0-65536 (2 byty). Identifikační číslo slouží společně s dalšími parametry k fragmentování paketů, a tak se každým odeslaným paketem mění. V operačního systému Linux se identifikační číslo mění náhodně a není možné ho předpovědět. U operačního systému Windows se identifikační číslo inkrementuje o přesně danou konstantu (velikost konstanty se liší mezi verzemi Windows) a je tedy možné předpovědět inkrementované identifikační číslo. Následující schéma 2.3 popisuje krokově TCP Idle skenování:



Obr. 2.3: Schéma TCP Idle skenování [7]

1. V prvním kroku útočník odešle na prostředníka TCP paket s příznaky SYN a ACK. Jelikož prostředník tato data neočekával, tak posílá zpátky TCP paket s příznakem RST. Důležitou informací je tu hodnota ID uvedená v IP hlavičce odeslaného RST paketu (ID=1).
2. Nyní útočník odešle oběti TCP paket s příznakem SYN na cílový port. Místo vlastní IP adresy však do pole odesílatele nastaví IP adresu prostředníka.
3. Oběť si logicky myslí, že prostředník chce pomocí TCP paketu SYN navázat spojení. Pokud je port otevřený, oběť pošle prostředníkovi TCP paket s příznaky SYN a ACK. Jakmile však prostředník obdrží TCP paket od serveru, odpoví TCP paketem s příznakem RST, jelikož si zahájení komunikace nepřál. V tomto kroku tedy prostředník zvyšuje hodnotu ID v IP hlavičce o n, v tomto případě o 1 (ID = 2). Pokud by byl port na oběti zavřený, oběť by nereagovala a komunikace mezi prostředníkem a serverem by neproběhla.
4. Poslední krok je stejný jako první. Zasláním SYN a ACK útočník získá odpověď v podobě TCP paketu s příznakem RST. Důležitá je zde koncová hodnota ID (v případě že komunikace mezi obětí a prostředníkem proběhla ID=3).

Na základě inkrementace hodnoty ID v IP hlavičce lze odvodit, že komunikace mezi obětí a serverem proběhla (v uvedeném případě ID = 3). Pokud komunikace neproběhla, bude hodnota ID = 2.

Je třeba si uvědomit, že se hodnota ID inkrementuje s každým odeslaným paketem prostředníka, tudíž je potřeba aby na prostředníkovi byl minimální síťový provoz a také, aby prostředník odpovídal na nečekaný TCP paket s příznaky SYN+ACK zasláním RST. Realizace tohoto typu útoku je opět možná pomocí programu nmap,

a to pomocí následujícího příkazu:

```
nmap -sI IP_ADRESA_PROSTREDNIKA IP_ADRESA_OBETI
```

Nmap provádí několikanásobné odesílání paketů tak, aby si oběť a prostředník vyměnili co nejvíce paketů. Tím se zvýší i počet inkrementací ID a lze rozhodnout o stavu portu, i když probíhá na prostředníkovi jiná komunikace [7].

UDP skenování (pouze UDP)

Protokol UDP je oproti TCP jednodušší. Jedná se o nespojovanou službu, nenavazuje se spojení. Strana, která odesílá UDP datagram nehlídá zda se datagram opravdu doručil. Využívá se převážně pro odeslání dat, u kterých příliš nezáleží na ztrátě, ale je požadováno co nejrychlejší doručení. UDP také méně zatěžuje linku.

Skenování UDP probíhá tak, že klient odesílá prázdné UDP pakety na určité porty. V případě že port není otevřený, server odešle ICMP paket s informací o nedostupnosti portu. Pokud je port otevřený, tak server odešle nějaká data klientovi, nebo příchozí data ignoruje a nepošle nic. Problémem je však fakt, že protokol ICMP je z důvodu bezpečnosti často filtrován. Tím se vytváří iluze, že port je uzavřený, ačkoli z důvodu filtrování nebyla ICMP zpráva doručena. Realizace UDP skenování je možná pomocí **nmap** příkazem:

```
nmap -sU -p 0-65536 (ROZSAH_PORTU) IP_ADRESA_OBETI.
```

Při použití příznaku `-sV` se bude **nmap** snažit získat více informací o portech na základě odesílání určitých dat na porty [7].

2.5 Výčet zranitelností

V momentě, kdy útočník vlastní dostatečné množství informací o cílovém serveru, přichází na řadu využití těchto informací k zjištění slabin systému (enumeration). V této části útočník provádí aktivní připojování na cílový server za účelem odhalení slabin. Jelikož se jedná o aktivní připojení, tak bude s největší pravděpodobností logováno cílovým serverem.

Obecně útočníci vyhledávají uživatelská jména (například pro následné využití v útocích na získání hesel), hledají špatně nakonfigurované sdílení prostředků (jako jsou například nezabezpečené sdílené soubory) a v neposlední řadě zastaralý software, který obsahuje známou zranitelnost (jako je například web server se zranitelností – přetečení na zásobníku). V momentě, kdy útočník takovouto zranitelnost odhalí, bývá pouze otázka času, než je tato slabina zneužita. Pro zjišťování konkrétních slabin systémů existují velké množství způsobů [5].

2.5.1 Zjišťování verzí

Zjištění verze běžící služby (anglicky service fingerprinting) je důležité pro odhalení slabin dané verze. Standardem pro tento typ skenování je velmi silný nástroj **nmap** [5].

Zjištění verze pomocí utility Nmap

Nmap lze využít k získání informací o verzi služeb běžících na cílových protokolech. Při využití parametru `-sV` lze získat podrobné informace o verzi protokolu cílového portu. Nmap zjišťuje verzi na základě známých odlišností mezi odpověďmi jednotlivých služeb. Informace o typu odpovědí má uloženy v souboru **nmap-service-probe**. Příkaz pro zjištění verze může vypadat následovně:

```
nmap -sV www.domena.cz -p cislo_portu
```

2.5.2 Skenování zranitelností

Programy pro automatické skenování obsahují pravidelně aktualizovanou databázi známých zranitelností a jejich popis. Zranitelnosti mohou být na úrovni operačních systémů, služeb či webových aplikací. Nevýhodou automatického skenování zranitelnosti je snadné odhalení ze strany serveru. V současné době existuje několik společností zabývající se vývojem komerčních aplikací, jako je McAfee, Qualy, Rapid7. Existuje také několik open source aplikací, jako je např. OpenVAS.

Nessus skener

Nessus je jedním z velmi známých skenerů zranitelností. Obsahuje grafické rozhraní a je jednoduchý na používání. Výhodou je podpora většiny používaných operačních systémů (včetně iOS a Android). Nessus také obsahuje možnost vytváření vlastních pluginů za pomoci jazyka NASL (Nessus Attack Scripting Language).

2.5.3 Zneužití uvítacích zpráv

Jedním ze základních způsobů získání informací o zranitelnosti je zneužití uvítacích správ serveru (anglicky banner grabbing). V momentě, kdy se útočník začne vzdáleně připojovat na určitý port serveru, je mu ve většině případů odeslána uvítací zpráva obsahující informace. Po prozkoumání takto získaných informací lze získat verze služeb běžících na serveru [5].

Netcat

Pro realizaci lze využít například program Netcat. Pro zjištění informací o verzi služby na portu 80 (HTTP) lze využít příkazu:

```
ncat www.domena.cz cislo_portu
```

2.5.4 Výčet zranitelností u běžných služeb

Spuštěné služby jsou velkou slabinou běžících stanic. Špatná konfigurace či zastaralá verze služby s chybou může mít fatální následky v případě odhalení a využití slabiny útočníkem.

DNS, TCP/UDP 53

Služba DNS pracuje nejčastěji na protokolu UDP portu 53, může však také běžet na TCP portu 53 z důvodu podpory rozšíření, jako je přenos zóny. Zjištění slabiny probíhá na základě špatně nakonfigurovaného DNS běžícího na TCP protokolu.

DNS servery využívají takzvaný „přenos zóny“ k výměně informací mezi sebou. Odesílány jsou soubory dané zóny, kde se nacházejí informace, jako je mapování adresy hosta na IP adresu.

V případě, že má server spuštěnu službu Microsoft DNS pro podporu Active Directory (AD), je velké riziko, že útočník bude schopný získat i více informací než jen mapování hosta na IP adresu. Microsoft DNS podporuje takzvaný SRV (Service record) záznam k snadnému zjištění dostupných služeb. Těmito službami mohou být například LDAP, FTP nebo WWW.

Při použití příkazu `nslookup` a následným zadáním cílového DNS serveru lze zjistit informace z přenosu zóny serveru. Dalším využitelným příkazem je příkaz:

```
ls -d dnsserver.cz
```

Jedná se o vypsání informací ze symbolického odkazu nebo adresáře, na základě kterého lze zjistit mnoho informací, jako jsou adresa a číslo portu služby Globální katalog (_gc._tcp), adresu kontroléru pro Kerberos autentizaci (_kerberos._tcp) nebo adresu LDAP serveru (_ldap._tcp).

Pro zjišťování slabin u DNS lze využít i několik aplikací, jako je například dnsenum `https://github.com/fwaeytens/dnsenum`. Aplikace zvládá provést celou řadu útoků, jako je například Google scraping nalezení dalších subdomén, bruteforcing, reverzní vyhledávání, WHOIS dotazy a další [5].

Finger, TCP/UDP 79

Jedním z nejstarších způsobů vniknutí na UNIX/Linux server je za pomocí využití utility zvané finger. Jedná se o utilitu sloužící ke zobrazení informací o uživateli na cílovém serveru. Ačkoli se jedná o velmi starou a známou slabinu, spousta současných serverů nemá tuto slabinu ošetřenou. Utilita finger běží na poortu 79. Názorný příklad zneužití je zobrazen níže při použití příkazu:

```
finger -l @domena.cz
```

Bude vypsána informace o uživatelském účtu. Informace zobrazené touto utilitou jsou získávány z informačních polí umístěných v `/etc/passwd`. Další velmi nebezpečnou informací, kterou finger poskytuje, je výpis přihlášených uživatelů a jejich doby nečinnosti. Na základě takto získaných informací lze provádět například sociální inženýrství.

NetBIOS Session, TCP 139/445

Jednou z největších slabin operačního systému Windows je zneužití prázdné relace (null session). V případě, že se chce uživatel připojit na sdílený síťový disk nebo chce použít síťovou tiskárnu apod., využívá protokolu (SMB) Microsoft's Server Message Block. Tento protokol má za úkol sdílení souborů a sdílení tiskáren.

SMB je přístupný pomocí aplikačního rozhraní API a je pomocí něho možné získat bohaté množství informací o cílovém operačním systému. Tato slabina je nazývána achillovou patou Windows.

Pro využití stačí útočníkovi využít příkaz **net use** (obvykle využívaný pro připojení síťového disku s následujícími parametry:

```
net use \\server\IPC\$ "" /u:""
```

Jedná se o anonymní připojení (null session relaci), kdy je uživatelské heslo i jméno ponecháno prázdné. IPC$ značí meziprocesorovou sdílenou komunikaci.

V případě úspěšného provedení, může uživatel získat informace o sdílených datech, skupinách, hodnotách klíčů uložených v registru apod. Realizace tohoto útoku je velice snadná a množství informací získaných jeho provedením je veliké. Z toho důvodu se jedná o velmi nebezpečnou slabinu a je třeba jí ošetřit buď nastavením filtrovacího pravidla na firewall pro porty 139/445, případně zamezením přístupu anonymním uživatelům [5].

3 ÚTOKY NA DŮVĚRNOST

Důvěrnost systému je schopnost udržet data v tajnosti před neoprávněnými uživateli. Přístup k datům mají pouze oprávněné osoby na základně přidělených práv. Cílem útoku je důvěrnost systému narušit, a tím například získat tajná data. Typickým příkladem narušení důvěrnosti je odposlouchávání.

3.1 Útoky na komunikaci

3.1.1 Odposlouchávání

MAC záplava (MAC flooding)

Jedním ze způsobů realizace odposlechu cílové stanice je možnost využití techniky MAC záplavy (MAC Flooding). Jedná se o techniku, která útočí na přepínač, ke kterému je oběť připojena.

Klasický směrovač pracuje na linkové vrstvě a data tedy adresuje podle MAC adres v hlavičkách rámců. Směrovač obsahuje paměť, která se nazývá CAM tabulka (Content Addressable Memory table). Účelem této paměti je ukládání záznamů o MAC adresách, které má přepínač připojen do portů.

Pokud je tedy přepínač poprvé připojen do sítě a nebyl nijak předkonfigurován, má prázdnou CAM tabulku. V momentě, kdy mu přijde na port rámec, přepínač si přiřadí zdrojovou MAC adresu k portu, ze kterého rámec přišel. Cílovou MAC adresu nemá, proto odešle rámec na všechny porty. V momentě, kdy cílový host odpoví na příchozí rámec, je opět uložena i jeho MAC adresa do CAM tabulky. Problém je však v tom, že CAM tabulka má pouze omezenou paměť a v případě nově příchozích informací o MAC adresách staré maže [16].

Výše uvedené skutečnosti využívá také útok MAC flooding, kdy útočník začne velmi rychle generovat rámce s náhodnou MAC adresou. Přepínač si informace ukládá do CAM tabulky a informace o starých MAC adresách připojených zařízení si již nepamatuje, jelikož má CAM tabulka omezenou paměť. Tímto je způsobeno, že přepínač začne posílat opět veškerou příchozí komunikaci na všechny porty a dá se říci, že je degradován na síťový prvek hub. Realizace MAC flooding útoku je velice jednoduchá, lze využít programu Dsniff a spustit příkaz:

```
macof -i SitovyAdapter
```

Tímto příkazem je zajištěno generování náhodných MAC adres na určený síťový adaptér (nejčastěji eth0).

Výsledná komunikace je v tomto momentě přístupná všem hostům, kteří jsou k tomuto síťovému prvku připojeni, a tedy i útočníkovi.

K samotnému odposlechu síťové karty je možné využít program Wireshark, který patří k velmi oblíbeným open source programům pro sniffing. Jedná se o velmi intuitivní program s grafickým prostředím, kde stačí vybrat cílové rozhraní, které chce uživatel odposlouchávat. Ke zpřehlednění je možné využít několik filtrů provozu a takto zachycená data lze uložit i pro pozdější analýzu [16].

Odcizení portu

Tato technika spočívá v odcizení portu oběti (port stealing). Jak již bylo řečeno u MAC flooding, v CAM tabulce je uložená informace, jaká MAC adresa patří k danému portu.

V případě, že útočník zná MAC adresu oběti, je možné vygenerovat rámec, který bude mít cílovou MAC adresu rovnou MAC adrese útočníka a zdrojovou MAC adresu s MAC adresou oběti. Přepínač na základě tohoto rámce předpokládá, že oběť se přepojila na jiný port, ten port, ze kterého přišel podvržený paket. Nyní bude všechna komunikace probíhající k cíli směrována útočníkovi. V tomto případě tedy není oběti odeslán jediný rámec a ze strany oběti se může tato situace jevit jako běžný výpadek. Pokud však oběť odešle jakékoli data, bude CAM tabulka obnovena, proto je důležité po dobu útoku neustále na přepínač odesílat upravené rámce.

K realizaci tohoto útoku je možné využít například **arpoison** s následujícími parametry:

```
arpoison -i SitovyAdapter -d IpUtocnik -s IpObeti -t MacUtocnik
-r MacObeti -w 1
```

K samotnému zachycení dat je opět možné jako v předešlém útoku „MAC záplava“ využít programu Wireshark [16].

4 ÚTOKY NA AUTENTIČNOST

Autentičnost je schopnost systému zajistit uložená či přenášená data proti neoprávněné modifikaci. Útoky na autentičnost spočívají ve snaze změnit přenášená data, modifikovat běžící operační systém nebo běžící službu. Na základě této modifikace jsou možné další činy, tím může být získání tajných informací, přesměrování na phishingovou stránku případně smazání data a způsobení výpadku služby.

Tyto typy útoků mohou také sloužit pouze jako metody pro provedení útoků na důvěrnost nebo dostupnost. Zařazeny jsou však do útoků na autentičnost, jelikož umožňují zmíněnou modifikaci systému či dat.

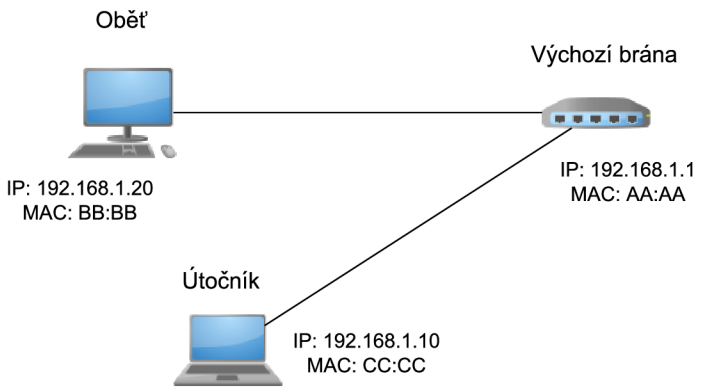
4.1 Útoky na komunikaci

4.1.1 ARP Spoofing

Jedná se o techniku Man in the middle (MITM). Útočník se dostává mezi komunikaci dvou koncových bodů.

Jak již bylo řečeno v části základní pojmy, ARP protokol funguje tak, že stanice vyšle žádost (ARP request) a stanice, které je tato zpráva určena odpoví (ARP response). Jelikož ARP je nestavový protokol, tak si stanice odesílající ARP request nezjišťuje, zda příchozí ARP response patří určitému requestu. Toto je hlavní zranitelnost, útočník tedy může odesílat falešné odpovědi (ARP response) bez žádostí a koncové stanice je přijímají a uloží si je do ARP tabulky [8].

K realizaci tohoto typu útoku je možné využít nástroje **arpspoof**, který je součástí balíčku **dsniff**, což je balíček, který obsahuje nástroje pro síťový audit a penetrační testování. Na níže uvedeném obrázku 4.1 je znázorněn praktický příklad ARP Spoofingu:



Obr. 4.1: Zapojení LAN pro ukázku útoku ARP spoofing

Protože tento typ útoku využívá přeposílání paketů skrze útočníka, je v první řadě nutné přepnout síťovou kartu tak, aby podporovala přeposílání. Toho lze v případě OS Linux docílit pomocí změny hodnoty: **/proc/sys/net/ipv4/ip_forward** z hodnoty `0` na hodnotu `1`. Ukázka příkazu:

```
echo '1' > /proc/sys/net/ipv4/ip_forward
```

Pokud bude útočník zasílat ARP response výchozí bráně 192.168.1.1 s informací, že jeho MAC adresa [CC:CC] má logickou adresu 192.168.1.20, dojde k aktualizaci ARP tabulky na výchozí bráně. Výchozí brána bude od této doby odesílat všechny pakety určené pro oběť útočníkovi. Výše uvedená technika je realizovatelná následujícím příkazem:

```
arpspoof [-i interface] [-t target] host
```

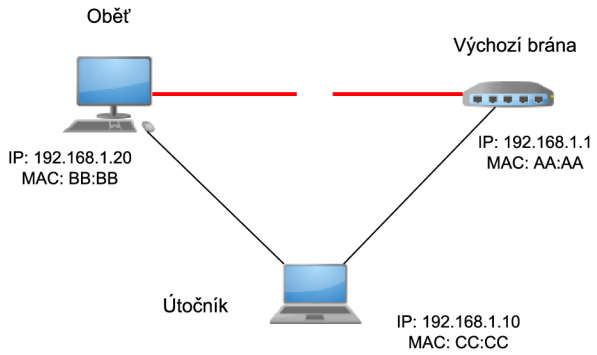
kde `-i` je síťové rozhraní, které je využito pro spoofing (v případě ethernetu např. eth0). Parametr `-t` specifikuje IP adresu cíle, kterému chceme zaslat falešnou ARP odpověď (v případě nevyplnění tohoto parametru bude odeslán všesměrově). Parametr `host` určuje adresu hostitele, kterému chceme odchytávat síťový provoz. Výsledný příkaz by v tomto případě byl:

```
arpspoof -i eth0 -t 192.168.1.1 192.168.1.20
```

Nyní je třeba přesvědčit hosta, že IP adresa útočníka je výchozí brána. To lze realizovat stejným způsobem jako předtím, tedy pomocí příkazu:

```
arpspoof -i eth0 -t 192.168.1.20 192.168.1.1
```

Nyní je útočník uprostřed komunikace mezi obětí a výchozí bránou. Může odchytávat, měnit či filtrovat probíhající komunikaci. Logická topologie se nyní změnila dle následujícího obrázku 4.2.



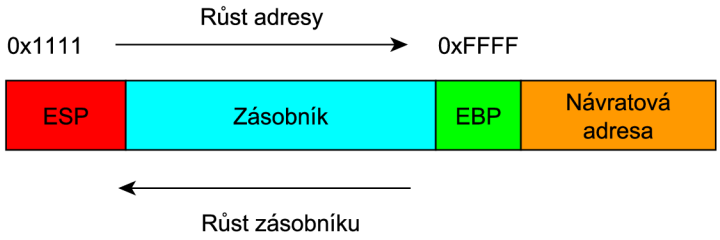
Obr. 4.2: Logická topologie po útoku ARP spoofing

4.2 Útoky na počítač a webovou aplikaci

4.2.1 Přetečení na zásobníku

Zásobník je datová struktura operačního systému. V paměti má každý proces svůj vlastní zásobník. Zásobník funguje na principu LIFO (Last-in, First-out). Nové objekty přibývají na vrchol zásobníku, ze kterého jsou i následně odebírány.

Při přidávání objektu na vrchol zásobníku se v assembleru využívá příkaz **PUSH**. Pro odebrání se využívá příkaz **POP**. O zásobník se starají dva registry, těmito registry jsou **EBP** (Extended base pointer) a **ESP** (Extended stack pointer). Registr EBP ukazuje na začátek zásobníku (vyšší adresa) a registr ESP na „vrchol“ zásobníku (nižší adresa) viz 4.3. Zásobník se naplňuje od vyšších adres k nižším [2].



Obr. 4.3: Rozložení paměti

Problém nastává v momentě, kdy se program pokusí zapsat více dat do zásobníku, než je jeho velikost. Pokud takováto situace nastane, může dojít ke zničení dat, pádu programu nebo ke spuštění škodlivého kódu. Je důležité říci, že buffer overflow nemusí vzniknout vždy cíleně „na základě pokusu“ útočníka, ale může nastat naprosto nepředvídatelně, např. při běhu programu.

Tato chyba se nejčastěji vyskytuje u kompilovaných programovacích jazyků, jako je např. C, C++ apod. Naopak jazyky interpretované, jako je například Java, Python, jsou vůči této chybě imunní. U interpretovaných jazyků však může chyba být přímo v interpretu [11]. Příklad chybně napsaného programu:

```
#include <string.h>
 void chybna_metoda ()
{
char  c[5];
strcpy(c, "123456789");
}
 int main (int argc, char **argv)
{
```

```
 chybna_metoda (argv[1]);
}
```

V metodě **chybna_metoda** je deklarován pěti-znakový řetězec, avšak v následujícím kroku se do něho pomocí metody **strcpy()** snažíme uložit řetězec s 9 znaky. Toto je typycký příklad buffer overflow. Takovýmto způsobem, může útočník přepsat například návratovou adresu a nahradit jí adresou na svojí část označovanou jako shellkód.

Shellkód je strojový kód, pomocí kterého lze získat např. přístup k systému. Je v binární podobě, zapsaný jako posloupnost šestnáctkových čísel. Na internetu se dají nalézt různé shellkódy pro všechny počítačové platformy [2].

4.2.2 Cros-Site scripting (XSS)

Jedná se o útok na webovou aplikaci. Zkratka XSS označuje Cross-site scripting, možné přeložit jako skriptování napříč stránkou. Tento typ zranitelnosti je známý již mnoho let, přesto se lze setkat s touto zranitelností ve více než v 80 % webových aplikací. Zranitelnost má několik podob, které jsou nejčastěji děleny na trvalé (persistentní) XSS, dočasné (Non-persistentní) XSS a Lokální (DOM-based) XSS.

Původ této zranitelnosti začal s využíváním Javascriptu ve webových stránkách za účelem oživení obsahu. V dřívější době bylo možné na webových stránkách zobrazovat pouze statické informace. Zobrazit aktuální datum, čas nebo přistupovat k objektům na stránce a měnit dynamicky jejich obsah bylo v dřívější době nemožné. S implementací v podobě Javascriptu přišla společnost Netscape. V současné době existují i další skriptovací jazyky, ale vývojáři využívají převážně Javascript.

Způsoby, jak vložit do stánky skripty, jsou tři. První možností je vložení skriptu mezi tagy `<SCRIPT>` a `</SCRIPT>`. Tyto znaky se dají vložit do hlavičky nebo těla HTML dokumentu. Poté, co prohlížeč narazí na tag `<SCRIPT>`, vykoná obsahující kód. Ukázka takto vloženého skriptu:

```
<script type="text/javascript">
  <!--
    document.write("Nazdar světe!");
  //-->
</script>
```

Skript vypíše na webovou stránku text „Nazdar světe!“. Uvnitř skriptu se nachází také HTML značky pro komentáře `<!-, //->` v případě, že by prohlížeč nepodporoval Javascript. V tomto případě by se komentovaná část prováděla jako HTML a skript by se na webové stránce nezobrazil. V dnešní době existuje jen málo webových prohlížečů, které neumí pracovat s Javascriptem [12].

Druhou možností je načtení Javascriptového kódu z externího souboru. Tento způsob má výhodu, jelikož v případě rozsáhlého Javascriptu se nestane kód stránky nepřehledným. Další výhodou je snadná možnost použití Javascriptového kódu na více webových stránkách a v případě modifikace skriptu stačí editovat pouze jeden soubor ***.js** místo webových stránek. Ukázka vložení kódu:

```
<script src="myscript.js"></script>
```

Poslední variantou je vložení pomocí In-line skriptu, který se vkládá jako atribut události. Tento níže uvedený skript se vykoná jako reakce na nějakou událost. Touto událostí může být kliknutí myší na odkaz, přejetí kurzorem myši apod.:

```
<a href="http://www.vutbr.cz" onmouseover="alert('Ahoj světe!')">
vutbr.cz</a>
```

Výše uvedený skript zobrazí vyskakovací okno s textem „Ahoj světe!“ v případě najetí kurzorem na odkaz `www.vutbr.cz`.

Lokální (DOM-based) XSS

Při práci se skriptovacím jazykem na straně klienta se využívá modelu DOM (Document Object Model). Tento model definuje jednotlivé objekty na webové stránce, jako jsou otevřená okna prohlížeče, odstavce textu, vložené rámy, obrázky, formuláře, tlačítka a další prvky vyskytující se v dokumentu. Model DOM má přesně stanovenou hierarchii a při přístupu k objektu je třeba projít celou touto hierarchií.

U Lokálního XSS je využito výše uvedeného modelu na straně klienta. Skript, který útočník vloží do webové stránky se neposílá na server a provádí se u klienta. Z tohoto důvodu se nazývá Lokální XSS [12].

Například stránka, která získává informace z parametru URI a dále tyto data zpracovává, se v případě neošetření stává zranitelnou. Na níže uvedeném příkladu je ukázka skriptu webové stránky, který je náchylný na Lokální XSS útok:

```
<script>
 var pos=document.URL.indexOf("color=")+6;
 document.write("Vaše barva je:
"+document.URL.substring(pos,document.URL.length));
</script>
```

Skript vyčte hodnotu proměnné **color**, která byla zadána do URI a zobrazí ji na stránce. V případě, že však útočník zadá do proměnné kód, jako je např. `<script>alert('Ahoj Světe!');</script>`, výsledná stránka bude obsahovat níže uvedený kód, který se po načtení ihned provede:

```
Vaše barva je: <b><script>alert("XSS");</script></b>
```

Ne-persistentní XSS

S touto zranitelností se lze setkat na webových stránkách, které zpětně zobrazují vložené parametry ze strany uživatele jako odezvu. Jsou to například vyhledávací pole, chybové hlášení (např. 404 stránka nenalezena). V případě vyhledávání uživatel vloží hledaný výraz „např. XYZ“ a webová stránka provede vyhledávání v databázi apod. a reaguje zobrazením výsledků. Například „Hledaný výraz XYZ nebyl nalezen“. Pokud tento vstup není správně ošetřen, může útočník místo hledaného výrazu vložit skript jako například `<script>alert('Ahoj světe!');</script>`. Část výsledné stránky poté může vypadat následovně:

```
Na váš hledaný výraz <b><script>alert("XSS");</script></b>
bylo nalezeno <b>10</b> výsledků.
```

Skript je jednorázově proveden, ale nezůstává na serveru, od toho název nepersistentní.

Persistentní XSS

Tento typ útoku je možné využít u webových stránek, kde je možnost vložení trvalého příspěvku na webovou stránku. V praxi to mohou být návštěvní knihy, komentáře apod. Jestliže vstup od uživatele není dostatečně kontrolován, případně upraven logikou na straně serveru, může dojít k vložení kódu do stránky natrvalo neboli „persistentně“. Vložený skript se poté spustí každému návštěvníkovi dané webové stránky. V níže uvedeném příkladu je znázorněn kód, který by se mohl zobrazovat pro ostatní uživatele [12]:

```
<b>Neo</b>: <br>Mů skript ukrytý v příspěvku
<script>alert('Ahoj světe!');</script>
```

Na straně klienta se tedy spustí vložený skript a v tomto případě dojde pouze k otevření vyskakovacího okna s textem „Ahoj světe!“. V reálném prostředí jsou však možnosti využití tohoto útoku značné, od přesměrování na phishingovou stránku až po ukradení identity např. v podobě získání informací z cookies.

4.2.3 SQL Injection

Jedná se o útok pracující na principu podvržení vstupních parametrů aplikační části webové stránky. Výsledkem správně zvolených parametrů bude spuštění SQL kódů nad databází, který není tvůrci webové stránky předpokládán.

Jedná se o velmi nebezpečný útok, při kterém se útočník může dostat k datům uložených v databázi, jako jsou uživatelská jména nebo hesla (s největší pravděpodobností hash hesla). V horším případě je schopný pomocí SQL příkazů manipulovat s daty nebo i vymazat celou databázi. Ačkoli je tento útok známý, existuje stále velké množství webů, které nejsou proti tomuto útoku ošetřeny [13].

Neošetření vstupního formuláře

U webové stránky je zranitelné místo „neošetřený vstup“ v podobě přihlašovacího formuláře. Na níže uvedeném obrázku 4.4 je znázorněn typický přihlašovací formulář, který autentizuje uživatele na základě uživatelského jména a hesla.

Obr. 4.4: Přihlašovací formulář

Programová logika tohoto ověření by mohla být následující:

```
jmeno = getRequestString("UzivJmeno");
heslo = getRequestString("Heslo");
sql =
"SELECT * FROM Users WHERE Jmeno ='"+jmeno+"' AND Pass ='"+heslo+"'"
```

V případě, že uživatel do položky **UzivJmeno** a **Heslo** vloží například `"A" = "A"`, tak jeho výraz bude mít vždy logický výsledek `TRUE`. Tímto vznikne neošetřená situace a databáze na dotaz odpoví navrácením celého sloupce **Users** z databáze.

Neošetření vstupu z URI

Druhým příkladem je zneužití dotazů zasílaných na server URI. Níže uvedený kód má za úkol navrátit vyžádaný článek z databáze podle hodnoty `id` článku [14].

```
sql = "select * from clanky where id = '$_GET["id"]'"
```

Zavolání této funkce z prohlížeče může vypadat například takto:

```
http://www.domena.cz/clanek.php?id=1
```

Pokud útočník modifikuje dotaz v URI například na:

```
http://www.domena.cz/clanek.php?id=1 and "A"="A".
```

je docíleno stejného efektu jako v předchozím příkladu. Parametr je vždy pravdivý a webová stránka vypíše všechny hodnoty ze sloupce **clanky**.

Pokud útočník zjistí výše uvedené slabiny, může využít například příkazu `DROP`, který slouží k vymazání tabulky. Tímto krokem dojde k ochromení webové stránky. Pro zjištění zranitelnosti SQL injection lze využít jednoduchý nástroj **Havij** pro automatické skenování stránek [13].

5 ÚTOKY NA DOSTUPNOST

Dostupnost systému je nezbytná pro zajištění možnosti přístupu k datům. Útoky typu DoS (Denial of Service) útočí na systém za účelem omezení dostupnosti. Existující DoS útoky lze rozdělit na dvě základní skupiny:

DoS útoky na komunikaci. Tyto útoky se snaží vytížit linku připojeného systému, systém poté není schopný přijmout relevantní žádosti uživatelů a stává se tak nedostupným.

DoS útoky na počítač. V tomto případě je cíleno na hardware či software operačního systému. Na základě chyby v implementaci síťového protokolu je možné vytížit cílový server do takové míry, že není schopný obsluhovat žádosti uživatelů, v horším případě dojde k jeho zhroucení.

5.1 DoS útoky na komunikaci

TCP záplavy (TCP flooding)

Jedná se o útoky založené na protokolu TCP. Existují záplavové útoky typu ACK, RST, FIN, NULL, URG, PSH, RST a podobné. Názvy výše uvedených útoků vychází z příznaků nastavených v TCP paketu.

UDP záplavy (UDP flooding)

Útok využívá bezstavový transportní protokol UDP. U tohoto útoku se využívá zranitelnosti u služeb echo a chargen. Tyto služby byly často puštěny na operačních systémech Linux nebo Windows. Služba echo měla za úkol příchozí pakety odeslat zpět na zdrojovou IP adresu (echo). Služba chargen při příchozím paketu odešle náhodná data na zdrojovou IP adresu.

Útok spočívá v odeslání UDP paketu s podvrženou zdrojovou IP adresou na server. V případě zapnuté služby echo je tento paket přeposlán na zdrojovou IP adresu a pokud touto adresou bude opět server s aktivní službou echo, vznikne nekonečná smyčka a servery si budou posílat pakety pořád dokola. Totéž platí i pro zapnutou službu chargen.

Výhodou tohoto útoku je fakt, že útočník je schopný zahltit linku serveru, který má mnohem rychlejší připojení. Výše uvedené služby se již na serverech většinou nepoužívají [9].

ICMP záplava

Útok využívající ICMP protokolu. Využívá pakety typu ICMP Echo. Tyto pakety jsou využívány k zjištění dostupnosti cílového serveru. Maximální velikost tohoto paketu je stanovena dle doporučení RFC na 548 B. Program ping pro Linux i Windows dovoluje odeslat mnohem větší velikost paketu, a to až 65 kB.

Princip ICMP echa je takový, že útočník odešle ICMP Echo request na cílový počítač a server mu zpátky odešle ICMP echo reply. Je přitom zachována původní velikost paketu. Tímto způsobem je tedy linka serveru zahlcena hned dvojnásobně jednak příchozím paketem a poté odesílaným paketem.

S výhodou lze opět využít podvržení zdrojové IP adresy na straně útočníka, a tím se vyhnout zahlcení linky v případě zasílaných odpovědí od serveru. Útok je velmi jednoduchý na realizaci, ve Windows lze využít program ping:[9]

```
ping „ip adresa" -t -l 65000
```

Ve Windows je maximální délka 65500 bitů. U Linuxu lze využít programu **hping3**.

Smurf

Jedná se o útok, který je principem podobný jako ICMP záplava. Oproti ICMP záplavě přidává k útoku i zesílení. Zesílení je realizováno pomocí zaslání pingu na IP adresu sítě a nastavením zdrojové IP adresy na IP adresu oběti.

Jelikož je ping odeslaný na adresu sítě, všechny počítače z cílové podsítě odpoví na zdrojovou adresu paketem ICMP Echo reply. Z toho lze odvodit, že velikost zesílení závisí na počtu počítačů v cílové síti. Realizace tohoto útoku je relativně jednoduchá pomocí programu **hping3**:

```
hping3 -1 --flood -a VICTIM_IP BROADCAST_ADDRESS
```

Na adrese `<http://www.powertech.no/smurf>` je možné nalézt databázi sítí, které mohou být k tomuto útoku využity. V současné době je již používání adres sítí v internetu převážně filtrováno. V současné době je tento typ útoku stále použitelný [9].

TTL Záplava (TTL Expiration flood)

Tento typ útoku využívá hodnotu TTL (Time to live) u IP protokolu. Každý odchozí paket má vygenerovanou hodnotu TTL, její hodnota bývá nastavena v rozmezí 64 až 255. V momentě, kdy se takovýto paket dostane na router (tedy zařízení pracující na třetí vrstvě TCP/IP modelu), je hodnota **n** nastaveného TTL snížena o **n-1**. Pokud nastane, že hodnota TTL se dostala na 0, je paket zahozen a zdrojové IP adrese

uvedené v IP paketu je odeslána zpráva o vypršení doby životnosti dat (vypršení TTL). Tato informace se nemusí dostat odesílateli, jelikož jsou často tyto informace filtrovány.

Princip útoku je takový, že útočník při odeslání paketu nastaví zdrojovou IP adresu na oběť. Poté odešle paket s nízkou hodnotou TTL a data pošle náhodnému cíli. Důsledkem toho TTL rychle vyprší a cíl pošle oběti zprávu o vypršení. U tohoto útoku je i malé zesílení, jelikož v případě odesílání ICMP paketu echo request s nízkou TTL má velikost 42 bytů. Zpráva zaslaná vypršením časovače má 70 bytů. Zesílení je tedy přibližně 1,7 násobné [9].

DNS zesilující útok (DNS amplification attack)

Útok je založený na posílání DNS dotazů. Pro realizaci tohoto útoku je potřeba veřejného relay DNS serveru. To je takový DNS server, který provede dohledání záznamu a je dostupný v internetu. Útočník nastaví zdrojovou IP adresu jako adresu oběti a pošle dotaz na DNS server. DNS server odešle na zdrojovou adresu odpověď. Výhodou tohoto útoku je zesílení, které je v běžném případě přibližně 6x větší. Velikost dotazu je průměrně 80 B, zatímco odpověď je 512 B.

Jako velmi zajímavé se jeví využití EDNS, což je rozšířené DNS, které umožňuje zasílat odpovědi větší než 4 KB. Tímto způsobem je docíleno až 50 násobného zesílení.

Takovéto zvětšení DNS záznamu je možné pomocí vložení dlouhého textového záznamu do DNS, které obvykle slouží jako komentář. Tento útok je považovaný za vysoce nebezpečný a v současné době realizovatelný [9].

Peer-to-peer útok

Poslední útok, který stojí jistě za zmínku, je útok pomocí P2P sítě. Jelikož jsou sítě P2P masivně využívané (podle jistého zdroje tvoří až 60 % internetového provozu) [10]. Takovýto útok je teoreticky možné realizovat s využitím protokolu Bitorrent, který slouží pro distribuci souborů pomocí P2P sítě. Celý systém Bitorrent se skládá ze čtyř hlavních částí:

Torrent soubor jedná se o soubor obsahující meta data popisující, jakým způsobem budou data zavedena při začátku stahování. Obvykle také obsahuje sekci „announce“, která specifikuje URL adresu trackeru, a sekci „info“, která popisuje názvy souborů, na které odkazuje, délku a SHA-1 otisk dat.

BitTorrent Klient program, který implementuje BitTorrent protokol. Umožňuje uživateli stahovat/nahrávat soubory popsané v torrent souboru a ty pak následně sdílet a stahovat od ostatních uživatelů. V síti BitTorrent existují dva

typy uživatelů. Seeders (rozsévači), jsou uživatelé, kteří mají již stažený celý soubor a „rozesévají“ ho ostatním uživatelům. Další uživatelé jsou leechers (pijavice), to jsou uživatelé, kteří nemají stažený celý soubor.

Webová stránka/Vyhledávač Torrent souborů stránky poskytující torrent soubory ke stažení.

Tracker jedná se o hosta, který je zodpovědný za koordinaci rozesílání cílového souboru. Sleduje klienty stahující cílový soubor pod určitým torrent souborem. Směruje nově příchozí uživatele na uživatele, kteří mají část (leechers) nebo celý soubor (seeders) [10].

Za normálních okolností uživatel navštíví webové stránky poskytující torrent soubory a cílový soubor si stáhne do počítače. Pomocí BitTorrent klienta uživatel torrent otevře. V tomto okamžiku klient vyhledá tracker (dle informací obsažených v torrent souboru). Tracker aplikaci sdělí adresy ostatních uživatelů připojených k torrentu, uživatel se tak může spojit přímo s uživateli (P2P) a stáhnout soubor.

V současné době nejsou striktně kontrolovány torrent soubory, které se nahrávají na webové stránky. Potenciální útok lze realizovat nastavením trackeru na IP adresu oběti v torrent souboru. Hosté, kteří mají v úmyslu stáhnout soubor, na který odkazuje torrent soubor, budou v pravidelných intervalech kontaktovat tracker, čímž lze docílit DDoS útoku.

Síla útoku stojí na počtu uživatelů, kteří mají o soubor zájem. Lze tedy využít sociálního inženýrství a vhodně zvolit název, jako je například název nově vycházejícího filmu, hry apod.

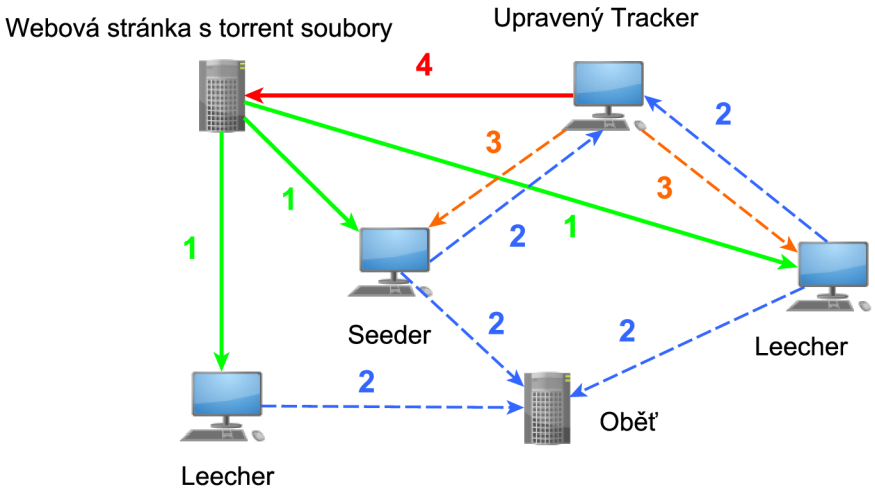
Nahráním takového souboru do vyhledávače torrentů (jako např. `thepiratebay.se>`) není zcela jisté, zda se dostane do prvních řebříčků vyhledávání nebo zda se vůbec na www stránkách objeví. Dnešní weby poskytující torrent soubory obsahují kontrolní mechanismy, které získávají statistické informace od trackeru a v případě, že počet uživatelů je nulový, soubor s torrentem nezobrazí.

Tento problém však lze řešit využitím funkce multi-tracker, který slouží k zvýšení dostupnosti torrentu. Útočník může spustit svůj modifikovaný tracker, který bude webovou stránku zásobovat falešnou statistickou informací o počtu přispěvatelů a v torrent souboru bude také IP adresa druhého trackeru, která bude nastavena na IP adresu oběti [10].

Na níže uvedeném schématu je znázorněn DDoS útok s využitím funkce multi-tracker:

1. Prvním krokem je stažení torrent souboru uživateli z webových stránek.
2. Nyní je kontaktován tracker uvedený v torrent souboru. Při využití multi-tracker funkce je zátěž rozkládána mezi více trackerů. Upravený tracker se tedy také stává obětí útoku DDoS.

3. Upravený tracker zasílá list uživatelů připojených k torrentu.
4. Upravený tracker zasílá falešné statistické informace webové stránce.



Obr. 5.1: Schéma DDoS útoku s využitím BitTorrent protokolu

5.2 DoS útoky na počítač

SYN záplava (SYN flood)

Tento typ útoku využívá trojcestný proces při navazování TCP spojení. Útočník pošle paket pro navázání TCP spojení s příznakem SYN na cílový server. Cílový server předpokládá, že se jedná o klasického klienta žádajícího o připojení, a proto alokuje systémové prostředky pro budoucí spojení a odešle potvrzovací paket s příznaky SYN a ACK útočníkovi. Jelikož je možné, že se odeslaný paket může ztratit, server vygeneruje čas, který bude čekat na přijetí potvrzovacího paketu ACK od útočníka. Tato odpověď však nikdy nepřijde. Výsledkem výše uvedeného je server s alokovanými prostředky čekající na odpověď, tomuto stavu se říká napůl-otevřené spojení.

Pokud na server nepřijde potvrzovací paket ACK, tak za normálních okolností, po uplynutí časovače, server uvolní alokované prostředky. Problém nastává ve chvíli, kdy útočník generuje větší množství SYN paketů a generuje je větší rychlostí, než je server schopný zahazovat expirované žádosti. Server v tomto případě není schopný přijímat nové spojení, dokud neuvolní prostředky. Spojení již vytvořená zůstávají funkční. Dalším možným scénářem je zhroucení systému na základě nedostatku prostředků.

Velkou výhodou tohoto útoku je možnost podvrhnutí falešné odchozí IP adresy, tím se stává útočník špatně dohledaný. Nepomůže tedy přidání příchozího pravidla na firewall, jelikož útočník může generovat IP adresy náhodně. Další výhodou je fakt, že útočník nepotřebuje k realizaci velkou šířku pásma. Není potřeba zahltit příchozí linku serveru, ale je útočeno na systémové prostředky jako takové.

V prvních okamžicích po objevení útoku bylo řešení v podobě zvětšení paměti a zkrácení času při čekání na odpověď klienta. Další možností bylo nastavení firewallu tak, aby filtroval špatně vygenerované externí IP adresy (jako je např. 127.0.0.0, 10.0.0.0 apod.). Později bylo vymyšleno řešení v podobě SYN cookie. Pro realizaci tohoto útoku je možné využít nástroje **hping3** a níže uvedeného příkazu: [9]

```
hping3 -S --flood -V www.hping3testsite.com
```

6 SOUBOR LABORATORNÍCH ÚLOH

Jedním z hlavních úkolů praktické části diplomové práce je vytvoření laboratorních úloh, které budou zaměřeny na problematiku bezpečnosti počítačů a počítačových sítí, kde si budou moci studenti prakticky vyzkoušet vybrané útoky.

6.1 Požadavky

Na laboratorní úlohy jsou kladeny následující požadavky:

- Úloha má být součástí počítačových cvičení.
- Cílem úloh je ověření realizovatelnosti popisovaných útoků.
- Úlohy budou prováděny pomocí virtuálních strojů na jediném počítači, na hostujících počítačích je OS Microsoft Windows 7 a předinstalovaný virtualizační nástroj VMware player.
- Čas na zpracování jedné úlohy je stanoven přibližně na 90 minut.
- Od studentů nebudou požadovány žádné výstupní protokoly.

6.2 Cíl laboratorních úloh

Studenti by si měli ověřit vybrané útoky v praxi, měli by získat základní povědomí o postupech při realizaci popisovaných útoků. Dále by se měli seznámit s programy využívanými pro realizaci samotných útoků.

6.3 Obsah laboratorních úloh

V této části jsou popsány jednotlivé laboratorní úlohy, jejich obsah a účel.

Pro každou laboratorní úlohu byl vytvořen text laboratorní úlohy a dokumentace pro vyučujícího, všechny dokumenty je možné nalézt na konci dokumentu v části přílohy. Byly také vytvořeny komentované videoukázky ke každému z popisovaných útoků, které je možné nalézt na přiloženém DVD.

6.3.1 Laboratorní úloha - Část 1.

V této úloze jsou studenti seznámeni se službou WWW a detailně s protokolem HTTP, s jeho metodami a principem komunikace. Dále jsou studenti seznámeni se síťovou analýzou, pojmem sniffing a programem **Wireshark**.

Praktická část slouží pro získání základních informací o síťovém analyzátoru Wireshark, používáním filtrů pro zachycenou komunikaci a analýzu protokolu HTTP.

Součástí úlohy je také samostatný úkol, který slouží jako ověření získaných znalostí. Student by měl být schopen ovládat program Wireshark, měl by se zorientovat v komunikaci pomocí protokolu HTTP a nalézt v něm požadované informace.

6.3.2 Laboratorní úloha - Část 2.

V teoretické části této úlohy jsou studenti seznámeni s ARP a DNS protokolem, dále jsou seznámeni s Linuxovou distribucí Kali linux určenou pro penetrační testování.

V praktické části si studenti otestují útok ARP spoofing použitím programu **arpspoof**, pomocí kterého se dostanou mezi komunikační kanál Klienta a Serveru. Následně proběhne odposlech komunikace pomocí programu Wireshark. Tímto by si studenti měli uvědomit, že se nacházejí uprostřed komunikačního kanálu (klient/-server).

Úloha následně obsahuje navazující útok DNS spoofing, kde si studenti vyzkouší zasílání falešných DNS odpovědí klientovi a přesměrování na IP adresu útočníka pomocí programu **dnsspoof**. Dále studenti spustí falešný HTTP server apache na útočníkovi a přesunou připravené soubory s falešnou webovou aplikací. Tím vytvoří jednoduchou formu phishingu a pomocí falešné stránky následně odchytí hesla zasílané klientem.

Součástí úlohy je také samostatný úkol, na kterém je ověřeno, zda student je student schopný modifikovat útok DNS spoofing, a tím ověřit pochopení principu a realizace útoku.

6.3.3 Laboratorní úloha - Část 3.

Ve třetí části laboratorní úlohy, jsou studenti seznámeni se základním principem HTTPS protokolu, dále jsou seznámeni s útoky na webovou aplikaci - Cross-site scripting (XSS) a SQL injection.

V první části praktické úlohy si studenti za pomoci programu **sslstrip** vyzkouší realizaci útoku SSL strip. Komunikace zde probíhá mezi klientem a vytvořenou webovou aplikací umístěnou na serveru. Na počítači útočníka je nejprve realizován ARP spoofing známý již z předchozí úlohy. Poté je spuštěn program sslstrip. Studenti si ověří úspěšnost útoku odchycením hesla zaslaného klientem.

V druhé části laboratorní úlohy, se studenti přihlásí do vytvořené webové aplikace, která obsahuje záměrně neošetřené vstupy. Na této aplikaci si vyzkouší realizaci několika základních technik XSS útoků (Ne-persistentní, persistentní, lokální) a SQL útoků.

Součástí úlohy je samostatný úkol, kde je ověřeno, zda student pochopil princip SQL útoku a zda je schopný modifikovat vstupní sql kód tak, aby modifikoval

výsledek útoku.

6.3.4 Laboratorní úloha - Část 4.

V poslední části laboratorní úlohy se studenti v rámci teorie seznámí s útoky typu DoS, s principem záplavového útoku, s principem útoku využívající chybu a jejich rozdílem. Dále jsou seznámeni s frameworkem **Metasploit**, s jeho principem, účelem a základním ovládáním.

V praktické části si studenti vyzkouší tři záplavové útoky (ICMP flood, TCP flood a UDP flood). U každého útoku mají možnost pozorovat statistiky zatížení linky serveru pomocí aplikace **nload** a prodloužení v čase načítání stránky pomocí nainstalovaného pluginu **app.telemetry Page Speed Monitor** v prohlížeči. Dále si studenti vyzkouší DoS útok, TCP reset využívající útoku ARP spoofing, který je realizován pomocí programu Ettercap.

Ve druhé části proběhne útok na klientskou stanici pomocí frameworku Metasploit. Pomoci tohoto útoku si studenti vyzkouší základní práci s frameworkem. Na stanici je nainstalovaná aplikace **vsftpd**, která obsahuje backdoor, pomocí něhož bude realizováno vniknutí do systému. Po vniknutí dojde k ověření práv přihlášeného uživatele a informací o cílovém systému.

Součástí úlohy je také úkol, ve kterém jsou studenti motivováni použít internet a vyhledat příkazy pro provedení jednoduché operace na klientovi skrze otevřené spojení pomocí backdooru.

7 POPIS VYTVOŘENÝCH VIRTUÁLNÍCH POČÍTAČŮ

Níže je popsána zvolená technologie, která byla využita pro realizaci vybraných útoků a slouží pro účely laboratorních úloh. Všechny virtuální počítače jsou k dispozici na přiloženém DVD.

7.1 Zvolené technologie

7.1.1 VMware Player

Pro virtualizaci byl vybrán nástroj VMware player. Z hlediska požadavků tento nástroj splňuje všechny požadované parametry potřebné pro realizaci úloh. Další výhodou je fakt, že je na hostujících počítačích již tento nástroj nainstalován.

WMware player je nástroj pro virtualizaci operačních systémů. Jedná se o freeware řešení od firmy VMware, která poskytuje i komerční řešení virtualizace desktopu v podobě VMware Workstation, serverová řešení VMware Server a mnoho dalších nástrojů ke komplexní virtualizaci.

Základní funkcí VMware playeru je virtualizace hardwaru a enkapsulace virtuálního systému. Virtualizací hardwaru je myšleno vytvoření kompletního virtuálního PC počínaje od CPU, pamětí RAM, grafických adaptérů až po síťové karty, řadiče disků apod. To znamená, že hostitelský počítač, na kterém je VMware spuštěn, je zcela odizolovaný Sandbox mechanismus od běžících virtualizovaných systémů a v případě pádu virtuálního systému není hostitelský systém ohrožen.

Oddělení hostitelského a virtuálního počítače přináší nespornou výhodu ve vyšší bezpečnosti a značnému usnadnění práce. Například při testování neznámých aplikací či pokusech s konfigurací operačního systému lze právě s výhodou využít tuto virtualizaci.

Pro využití programu VMware player jsou stanoveny požadavky na hostující PC. Player je možné spustit jak na x86 architektuře, tak na 64-bit s následujícími požadavky:

- Procesor 1,3 Ghz (V případě x64 verze je pro AMD je nutná podpora segment-limit = long mode, pro Intel VT-x).
- Operační paměť 1GB minimum (Doporučené 2 GB).

Všechny výše uvedené požadavky budoucí hostující počítače splňují.

Pro klientský počítač byl nainstalován také VMware Tools, což je balíček ovladačů pro vylepšení grafického výkonu, vylepšení propojení hostitelského a hostujícího

systému, podpora plug-and-play, synchronizace času a další. Tento balíček se instaluje po instalaci operačního systému ve virtuálním stroji a je možné ho nainstalovat ve většině dnes používaných desktopových prostředí: Windows, Linux, FeeeBSD [17].

7.1.2 Kali Linux

Kali Linux je Linuxová distribuce společnosti Offensive-security, která vyšla v roce 2013 a je určena pro pokročilé penetrační testování a realizaci bezpečnostních auditů. Tato distribuce je následovníkem Linuxové distribuce Black Track 5 a byla vytvořena stejnou společností.

Minimální konfigurace pro běh Kali Linuxu na architektuře i386 či amd64 je 1 Ghz CPU, 8 GB HDD a 300 MB RAM. Kromě i386 a amd64 je podporováno několik různých ARM platforem, jako je rk3306/ss808, Raspberry Pi, ODROID U2/X2, Samsung Chromebook či Galaxy Note 10.1 [15].

Kali Linux je postaven na základě OS Debian (verze 7.0 „Wheezly“) a je tedy kompatibilní s **.deb** balíčky. V základní verzi je podporovnáno více jak 300 nástrojů pro penetrační testování. Níže jsou jedny z nejznámějších uvedené:

Aircrack-ng
: Jedná se o nástroj pro testování zabezpečení Wi-Fi sítí. Nejčastěji se využívá k prolomení hesla do bezdrátové sítě zabezpečené pomocí WEP nebo WPA-PSK (nadstavbou tohoto nástroje je program **Wifite**, což je velmi jednoduchý program pro získání hesla z Wi-Fi sítí používající WEP, a **Fern Wi-Fi Cracker**, který se na základě brute force slovníkového útoku snaží zjistit WPA klíč).

Dsniff
: Jedná se o sniffer sloužící pro zachytávání mnoha protokolů. Automaticky analyzuje aplikační protokoly a poté ukládá informačně zajímavé části jako jsou vyplněné formuláře nebo hesla (součástí je i **Arpspoof**, který slouží k realizaci MITM skrze ARP protokol).

SSLstrip
: Program zachytává odchozí HTTP komunikaci oběti a přepisuje HTTPS odkazy na HTTP, tím nutí oběť k používání protokolu HTTP, se serverem je komunikace realizována pomocí HTTPS. Slouží k odchycení zasílaných dat.

Ettercap
: Jedná se o LAN sniffer a zároveň velmi silný nástroj pro MiTm útoky (plugin **remote_browser** dovoluje například sledovat WWW stránky, na které oběť surfuje).

Metasploit
: Open source framework pro penetrační testování, obsahuje informace o zná-

mých slabinách, obsahuje kódy využitelné po vniknutí do zranitelného systému (payload). Poskytuje možnost využít šifrovací techniky k odhalení payloadu pomocí IPS (Intrusion-prevention System) [15].

7.2 Nastavení virtuálního PC - Klient

V této části je popsáno nastavení provedené na klientském počítači s názvem **Klient - Xubuntu 14.04**.

7.2.1 Popis počítače a operačního systému

Parametry virtuálního počítače jsou následující: **Jednojádrový** procesor, operační paměť **1 GB**, virtuální pevný disk o maximální kapacitě **20 GB** (dynamicky se rozšiřující disk). Počítač je zapojen do virtuální sítě Custom: **VMnet1** (Host-only). Tato síť je oddělena z důvodu bezpečnosti od sítě na hostujícím počítači. Centrální prvek je zde **virtual network switch** [18].

Jako operační systém klienta byl zvolen systém Xubuntu ve verzi 14.04. Jedná se o svobodný a komunitně vyvíjený operační systém s licencí GNU/Linux, který vychází z distribuce Ubuntu. Tento systém byl zvolen na základě nízkých požadavků na počítač, kde je uváděná minimální operační paměť 512MB, grafické prostředí Xfce použité v Xubuntu je rovněž optimalizováno pro nízké nároky na cílový systém. Uživatelská jména a hesla do systému jsou uvedena v tabulce 7.1.

Virtuální počítač je spouštěn ze zachyceného **snapshotu:** `klient-default` s přesným nastavením. V momentě vypnutí virtuálního počítače se počítač navrátí zpět do bodu vytvoření uvedeného snapshotu. Toto nastavení bylo provedeno z důvodu zajištění opakovatelnosti laboratorních úloh. Jelikož VMware Player ve volné verzi neumožňuje práci se snapshoty [19], byly snapshoty vytvořeny ve vyšší verzi - VMware Workstation.

Tab. 7.1: Přístupové údaje - Xubuntu (klient)

Uživatelské jméno	**heslo**
metjuf	xubuntu
root	xubuntu

7.2.2 Nastavení sítě

Ačkoli virtuální síť VMnet1 disponuje DHCP serverem [18], bylo zvoleno statické nastavení IP adres z důvodu snadné orientace mezi počítači. Nastavení sítě na virtuálním počítači klienta je zobrazeno v následující tabulce 7.2:

Tab. 7.2: Statické nastavení sítě - Klient

IP adresa	192.168.1.10
Maska sítě	255.255.255.0
DNS server	192.168.1.12

Nastavení bylo provedeno editací souboru **/etc/network/interfaces**.

7.2.3 Instalované programy

Jelikož je v rámci první laboratorní úlohy třeba analyzovat síťový provoz, tak byl na klientský počítač nainstalován síťový analyzátor Wireshark ve verzi 1.10.6 pomocí programu apt-get příkazem:

```
sudo apt-get install wireshark
```

Ve čtvrté úloze probíhá útok na operační systém klienta. K tomuto účelu byl vybrán FTP server vsftpd obsahující backdoor. U verze 2.3.4 vsftpd serveru se objevila i podvržená verze archivu **vsftpd-2.3.4.tar.gz**, která obsahovala backdoor.

Jelikož bylo obtížné verzi obsahující backdoor nalézt pomocí oficiálních cest, tak byla stažena z následujícího odkazu: docs.google(not-so-vsftpd-2.3.4.tar.gz), následně byl získám oficiálně zveřejněný otisk **sha256** souboru, který obsahoval backdoor: [20]

```
2a4bb16562e0d594c37b4dd3b426cb012aa8457151d4718a5abd226cef9be3a5}
```

Stažený soubor byl ověřen oproti výše uvedenému otisku. Otisky se shodovaly a aplikace byla nainstalována. Instalace proběhla překonvertováním balíku `tar.gz` na instalační balík `deb` za pomoci programu **alien**:

```
sudo alien -k not-so-vsftpd-2.3.4.tar.gz
```

a následně nainstalována pomocí aplikace **gdebi**:

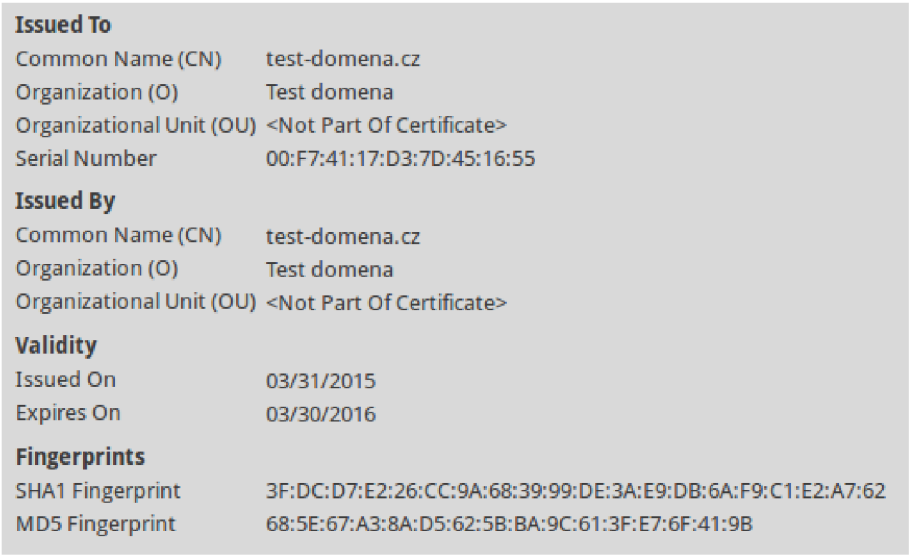
```
sudo gdebi not-so-vsftpd-2.3.4.deb
```

Na klientském počítači se také nachází FTP klient **FileZilla**, který slouží k nahrávání souborů (webové stránky) na server. Instalován byl pomocí příkazu:

```
sudo apt-get install filezilla
```

7.2.4 Nastavení realizovaná v prohlížeči (Mozilla Firefox)

Pro účely testování HTTPS komunikace byla přidána výjimka pro certifikát vygenerovaný a podepsaný serverem (viz obr. 7.1) pro přístup na zabezpečenou verzi webové aplikace `https://www.test-domena.cz`.



Obr. 7.1: Zasílaný certifikát

Ve čtvrté laboratorní úloze probíhá realizace útoků DoS. Pro přesnější výsledky byl do prohlížeče instalován doplněk pro měření rychlosti načítání stránek **app.telemetry Page Speed Monitor**. Tento doplňek je dostupný z následující stránky: addons.mozilla.org.

7.3 Nastavení virtuálního PC - Útočník

V této části je popsáno nastavení provedené na klientském počítači s názvem **Útočník - Kali Linux 1.1.0**.

7.3.1 Popis počítače a operačního systému

Parametry virtuálního počítače jsou následující: **Jednojádrový** procesor, operační paměť **1 GB**, virtuální pevný disk o maximální kapacitě **20 GB** (dynamicky se rozšiřující disk). Počítač je zapojen stejně jako klient do virtuální sítě Custom: **VMnet1** (Host-only). Tato síť je také oddělena od sítě hostujícího počítače. Centrálním prvkem je **virtual network switch** [18].

Operačním systémem pro útočníka byl zvolen systém Kali Linux verze 1.1.0, který byl již popsán v kapitole Kali Linux 7.1.2. Jedná se tedy o linuxovou distribuci společnosti Offensive-security, která vyšla v roce 2013, a je určená pro pokročilé penetrační testování a realizaci bezpečnostních auditů. V základní verzi tohoto systému je podporováno více jak 300 nástrojů pro penetrační testování. Uživatelské jméno a heslo do sytému je uvedeno v tabulce 7.3.

Virtuální počítač je spouštěn opět jako u klienta ze zachyceného **snapshotu:** `utocnik-default` s přesným nastavením. V momentě vypnutí virtuálního počítače se počítač taktéž navrátí zpět do bodu vytvoření uvedeného snapshotu.

Tab. 7.3: Přístupové údaje - Kali Linux (útočník)

Uživatelské jméno	**heslo**
root	kali

7.3.2 Nastavení sítě

Nastavení sítě bylo opět nastaveno staticky nastavením IP adres stejně jako v případě klienta. Nastavení sítě na virtuálním počítači útočníka je následující:

Tab. 7.4: Statické nastavení sítě - Útočník

IP adresa	192.168.1.11
Maska sítě	255.255.255.0

Nastavení bylo provedeno editací souboru **/etc/network/interfaces**.

7.3.3 Úprava souboru hosts pro SSLstrip

Ve třetí laboratorní úloze je realizován útok SSLstrip. Jelikož program **sslstrip** využívá ke komunikaci se serverem doménových jmen, byl přidán statický záznam do souboru hosts. Vypsaný konfigurační soubor je na obr 7.2.

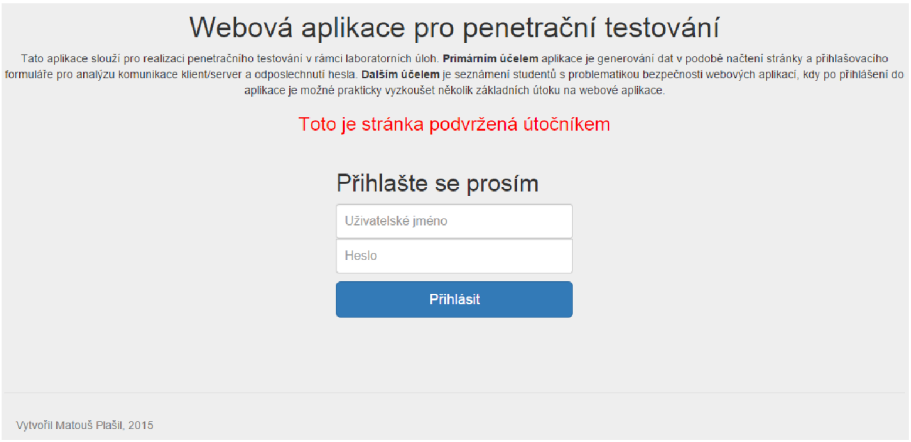
```
root@kali:~# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali.kali       kali
192.168.1.12    www.test-domena.cz
```

Obr. 7.2: Vypsaný obsah souboru hosts

7.3.4 Vytvoření webové stránky pro realizaci phishingu

V rámci druhé laboratorní úlohy je realizován jednoduchý phishing pomoci falešné webové stránky umístěné na útočníkovi.

Pro tento účel byla vytvořena modifikovaná kopie webové stránky, která je umístěna na ploše uživatele root v systému (`/root/Desktop/dnsSpoof`). Webová stránka se skládá ze souboru **index.php**, souboru **data.txt** a složky **css**, která obsahuje kaskádové styly. Při tvorbě kaskádových stylů byl využit open source framework Bootstrap verze 3.3.4. Soubor index.php obsahuje kopii přihlašovací stránky z adresy www.test-domena.cz v rámci virtuální sítě, viz obr. 7.3.



Obr. 7.3: Phishingová stránka

V okamžiku, kdy uživatel vyplní uživatelské jméno a heslo, dojde k odeslání těchto dat na útočníkův počítač. Zde je spuštěn php kód, který uloží zaslané uživatelské jméno a heslo do souboru data.txt. Uživatelské jméno a heslo je odděleno pomlčkou, následně je proveden skok na další řádek v souboru. Vkládaná data jsou připisována, soubor tedy není přepisován a je možné uložit libovolné množství uživatelských jmen a hesel viz níže uvedený kód:

```
<?php
if(isset($_POST['username']) && isset($_POST['password'])) {
//vytvoření proměnné pro zapsání (username-password)
  $data = $_POST['username'] . '-' . $_POST['password'] . "\n";
//zapsání proměnné do souboru data.txt
  $ret = file_put_contents('data.txt', $data, FILE_APPEND | LOCK_EX);
  }
?>
```

7.4 Nastavení virtuálního PC - Server

V této části je popsáno nastavení provedené na klientském počítači s názvem **Server - Ubuntu Server 14.04.2 LTS**

7.4.1 Popis počítače a operačního systému

Parametry virtuálního počítače jsou následující: **Jednojádrový** procesor, operační paměť **1 GB**, virtuální pevný disk o maximální kapacitě **20 GB** (dynamicky se rozšiřující disk). Počítač je zapojen stejně jako předchozí počítače do virtuální sítě Custom: **VMnet1** (Host-only). Tato síť je také oddělena od sítě hostujícího počítače. Centrálním prvkem je **virtual network switch** [18].

Operační systém pro server byl zvolen Ubuntu Server 14.04.2 LTS. Jedná se o operační systém určený pro provoz na serverech. Operační systém neobsahuje grafické rozhraní a je ovládán pouze pomocí příkazového řádku. Uživatelské jméno a heslo do sytému je uvedeno v tabulce 7.5.

Virtuální počítač je spouštěn opět jako u předchozích ze zachyceného **snapshotu:** `server-default` s přesným nastavením. V momentě vypnutí virtuálního počítače se počítač taktéž navrátí zpět do bodu vytvoření uvedeného snapshotu.

Tab. 7.5: Přístupové údaje - Ubuntu (Server)

Uživatelské jméno	**heslo**
root	ubuntu

7.4.2 Nastavení sítě

Nastavení sítě bylo opět provedeno pomocí statických IP adres stejně jako v případě předchozích dvou počítačů. Nastavení sítě na virtuálním počítači serveru je následující:

Tab. 7.6: Statické nastavení sítě - Server

IP adresa	192.168.1.12
Maska sítě	255.255.255.0

Nastavení bylo provedeno editací souboru **/etc/network/interfaces**.

7.4.3 Instalace DNS serveru (BIND9)

Pro potřeby laboratorních úloh byl na server nainstalován a nakonfigurován DNS server BIND9 (Berkler Internet Name Domain 9). Instalace serveru BIND9 proběhla pomocí příkazu:

```
apt-get install bind9 bind9utils bind9-doc
```

Dále proběhla základní konfigurace konfiguračního souboru **named.conf.options**, který se nachází v adresáři `/etc/bind/`. Zde proběhlo demonstrativní nastavení forwarders na Google DNS servery (pro přeposílání DNS dotazů serverem). Provedené nastavení:

```
forwarders {
8.8.8.8;
8.8.4.4;
};
```

Poté byl nakonfigurován soubor pro nastavení jednotlivých zón [21]. `Type master` určuje, že se jedná o primární DNS server. **file** určuje cestu k souboru s konfigurací dané zóny. Byla vytvořena jedna primární zóna `www.test-domena.cz`. Provedené nastavení:

```
zone "www.test-domena.cz"
{
type master;
file "/etc/bind/zones/db.test-domena.cz";
};
```

Poslední provedenou konfigurací byla konfigurace zóny [21] vytvořené výše. Nastavení zóny je uloženo v souboru **db.test-domena.cz**, která se nachází v souboru `/etc/bind/zones/`. Provedená konfigurace:

```
$TTL 86400
@ IN SOA test-domena.cz. root.test-domena.cz. (
                     1      ;Serial
                604800      ;Refresh
                 86400      ;Retry
               2419200      ;Expire
                 86400 )    ;Negative Cache TTL
;
       IN A        192.168.1.12
www    IN CNAME    test-domena.cz.
@      IN NS       test-domena.cz.
```

7.4.4 Instalace HTTP serveru (apache2) a podpory PHP

Na serveru byl nainstalován HTTP server apache2 ve verzi 2.4.7:

```
apt-get install apache2
```

Společně s apache2 byla nainstalována podpora pro skriptovací jazyk PHP verze 5.5.9 pomocí příkazu:

```
apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

7.4.5 Vygenerování a podepsání certifikátu (OpenSSL)

Pro účely testování HTTPS přenosu byl nainstalován program openssl pro vytváření a podepisování certifikátů pomocí příkazu:

```
apt-get install openssl
```

Následně proběhlo spuštění modulu ssl pro HTTP server apache2 pomocí příkazu:

```
sudo a2enmod ssl
```

Dále byla vytvořena složka pro umístění certifikátů, které budou generovány:

```
sudo mkdir /etc/apache2/ssl
```

Následně byl vytvořen privátní klíč **apache.key** a certifikát **apache.crt** do výše uvedené složky pomocí níže uvedeného příkazu:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
```

Popis jednotlivých parametrů: [22]

-req Žádost o podepsání certifikátu samotným serverem (tento parametr navazuje na parametr **-x509**)

-x509 Určení, že se jedná o certifikát, který bude podepsaný. Nikoli o pouhé vygenerování žádosti.

-nodes Soukromý klíč nebude chráněn heslem

-days 365 Platnost certifikátu

-newkey rsa:2048 Délka soukromého klíče

-keyout parametr určující výstup se soukromým klíčem

-out parametr určující výstup s certifikátem

Parametry vygenerovaného certifikátu:

```
Country Name:CZ
State or Province Name:Czech Republic
Locality Name:Brno
Organization Name:Test domena
Common Name:test-domena.cz
Email Address:test@domena.cz
```

Pro zprovoznění zabezpečené (HTTPS) verze webové aplikace bylo využito možnosti virtuálních hostů u Apache serveru [22]. V rámci nastavení byla určena cesta k souborům virtuálního hosta **DocumentRoot**, dále byly určeny cesty k soukromému klíči a podepsanému certifikátu SSLCertificateFile, SSLCertificateKeyFile. Konfigurace byla provedena modifikací souboru default-ssl.conf ve složce **/etc/apache2/sites-available/**. Obsah části konfiguračního souboru:

```
<IfModule mod_ssl.c>
    <VirtualHost _default_:443>
 DocumentRoot /var/www/html

SSLCertificateFile /etc/apache2/ssl/apache.pem
    SSLCertificateKeyFile /etc/apache2/ssl/apache.key
</VirtualHost>
</IfModule>
```

V poslední části byl SSL virtuální host povolen pomocí příkazu:

```
sudo a2ensite default-ssl.conf
```

7.4.6 Instalace aplikace nload

Pro účely měření vytížení síťového rozhraní eth0 serveru byl nainstalován program nload. Tento program monitoruje zatížení linky a je využíván ve čtvrté laboratorní úloze při realizaci DOS útoků. Nainstalován byl příkazem:

```
apt-get install nload
```

7.4.7 Vytvoření databáze (MySQL)

Pro potřeby webové aplikace a simulace SQL injection útoků ve čtvrté laboratorní úloze byl nainstalován MySQL server pomocí příkazu:

```
apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
```

Přístupové údaje do databáze jsou následující: **uživatelské jméno:** `root`, **heslo:** `ubuntu`.

Dále byla vytvořena databáze s názvem **test**, která obsahuje tabulku **members**. Struktura tabulky je následující:

Tab. 7.7: Struktura tabulky members

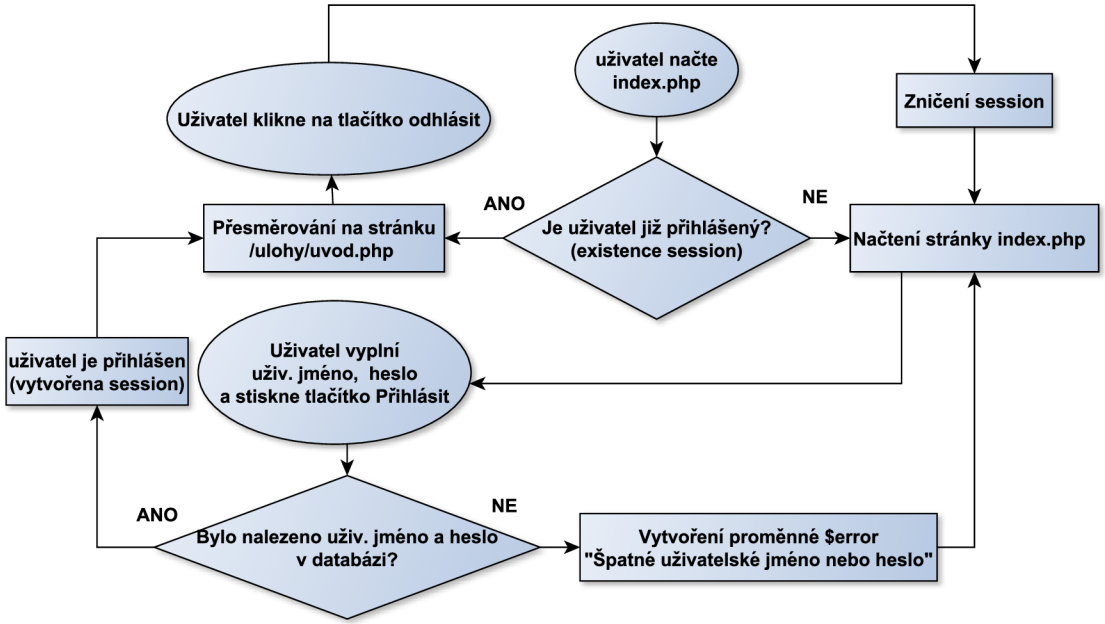
Název	**Typ**	**Další**
id	int(4)	AUTO_INCREMENT
username	varchar(65)	
password	varchar(65)	

7.4.8 Popis webové aplikace na serveru

Pro testovací účely byla vytvořena webová aplikace. Tato aplikace je umístěná na serveru v adresáři **/var/www/html/** a je dostupná z virtuálního stroje Klienta i Útočníka pod adresou `www.test-domena.cz`. Dále se také nachází v příloze ve složce **Webová aplikace - server** na přiloženém DVD.

Při tvorbě stránek bylo využito jazyka PHP, Javascript, HTML, MySQL a kaskádových stylů. Pro nastavení vzhledu aplikace byl využit framework Bootstrap v3.3.4.

Základní část webových stránek se skládá z následujících 4 php souborů, **index.php**, **login.php**, **logout.php**, **session.php**. Tyto soubory společně s databází kontrolují proceduru přihlašování a odhlašování. Princip procedury je na uvedeném diagramu 7.4.



Obr. 7.4: Procedura přihlášení a odhlášení

Soubor **index.php**, obsahuje přihlašovací stránku (viz obr. 7.5), a také níže uvedený PHP kód. Tento kód ověřuje existenci session. V případě, že session existuje, je uživatel automaticky přesměrován na úvodní stránku zobrazovanou po přihlášení **/ulohy/uvod.php**. Ve skriptu je také využita funkce include pro zahrnutí skriptu pro přihlašování login.php:

```
include('login.php'); // vlozeni login.php skriptu

if(isset($_SESSION['login_user'])){
header("location: ulohy/uvod.php");
}
```

Obr. 7.5: Přihlašovací obrazovka - login.php

Skript **login.php** má za úkol ověřit existenci uživatele v databázi **test**. V této databázi se nachází tabulka **members**, která obsahuje jednoho uživatele s uživatelským **jménem** `John` a **heslem** `1234`. (Pozn.: Heslo není uloženo jako otisk, jelikož se jedná pouze o testovací aplikaci v uzavřené testovací síti. V reálném provozu by bylo nutné ukládat otisk hesla pomocí hašovací funkce!)

V první části kódu je ověřováno, zda je uživatelské jméno a heslo vyplněné. Dále následuje uložení zaslaných informací do proměnných: uživ. jméno do `username` a heslo do `password`.

V tomto okamžiku je vytvořeno spojení s databází, dále je využito funkcí `stripslashes()` a `mysqli_real_escape_string()` jako ochrana proti útoku SQL injection.

Po této části následuje samotné vytvoření dotazu na databázi (viz níže uvedený PHP kód) a ověření, zda byl uživatel se zadaným jménem a heslem nalezen. V případě, že uživatel byl nalezen, je vytvořena session pro daného uživatele. V opačném případě je navrácena proměnná `error`, která obsahuje chybovou hlášku: `Špatné uživatelské jméno nebo heslo`. Níže je uveden kód:

```
$username = stripslashes($username);
$password = stripslashes($password);
$username = mysqli_real_escape_string($connection, $username);
$password = mysqli_real_escape_string($connection, $password);
// vyber databaze
$db = mysqli_select_db($connection, "test");
// SQL dotaz pro nalezeni uzivatele dle jmena a hesla
$query = mysqli_query($connection, "select␣*␣from␣members
␣where␣password='$password'␣AND␣username='$username'");
$rows = mysqli_num_rows($query);

if ($rows == 1) { //pokud byl uzivatel nalezen
$_SESSION['login_user']=$username; // zacatek session

} else { //pokud nebyl uzivatel nalezen
$error = "Spatne␣uzivatelske␣jmeno␣nebo␣heslo";
}
mysqli_close($connection); // uzavreni spojeni mysql
```

Dalším je skript **session.php**, který je umístěn na každé webové stránce, kde je nutné ověřit, zda je uživatel přihlášený. V tomto skriptu je připojena databáze **test**. Následně je získána hodnota uživatelského jména ze `session ID` uloženém v cookies prohlížeče. Následně dojde k vytvoření mysql dotazu, zda se získané uživatelské jméno nachází v databázi. V případě, že uživatelské jméno nebylo nalezeno, je uži-

vatel přesměrován na úvodní stránku **index.php**. V opačném případě je načtena požadovaná stránka. Níže je uvedená část popisovaného skriptu session.php:

```
$user_check=$_SESSION['login_user'];
// sql dotaz pro nalezeni uzivatele dle dane session
$ses_sql=mysqli_query($connection,
"select username from members where username='$user_check'");
$row = mysqli_fetch_assoc($ses_sql);
$login_session =$row['username'];
//pokud neni session nalezena
if(!isset($login_session)){
//uzavreni mysql spojeni
mysql_close($connection);
// presmerovani na stranku index.php
header('Location: index.php');
}
```

Skript **logout.php** je spuštěn, pokud uživatel po přihlášení klikne na tlačítko odhlásit. Jedná se o jednoduchý skript, který má za účel zničit aktuální session uživatele a následně dojde k přesměrování na stránku index.php, viz níže uvedený PHP kód:

```
session_start();
// pokud existuje session, dojde k jejimu zruseni
if(session_destroy()){
// Presmerovani na prihlasovaci stranku
header("Location: index.php");}
```

Testovací stránky pro XSS a SQL injection

Po přihlášení, které je popsáno výše, je načtena úvodní stránka **/ulohy/uvod.php**. Na stránce se nacházejí dva odkazy na dvě laboratorní úlohy: LAB1 - Cross-site scripting (XSS) a LAB2 - SQL injection, viz obr. 7.6.

Obr. 7.6: Úvodní obrazovka - uvod.php

První Laboratorní úloha LAB1 - Cross-site scripting (XSS) se skládá ze 4 souborů: **lab1.php**, **lab1_komentar.txt**, **lab1_save.php**, **lab1start.php**. V momentě, kdy uživatel klikne na odkaz první laboratorní úlohy LAB1, dojde ke spuštění skriptu **lab1start.php**.

Tento skript slouží pro inicializaci úlohy. V první části skriptu je spuštěna funkce `file_put_contents("lab1_komentar.txt", "")`, která slouží pro vymazání obsahu **lab1_komentar.txt**, ve kterém mohou být uloženy komentáře z předchozího testování. Poté dojde k přesměrování na stránku **lab1.php** s parametrem `jmeno=John`.

Soubor **lab1.php** obsahuje HTML kód pro vykreslení stránky. V první části souboru se nachází PHP skript pro testování ne-persistentního XSS. Skript má za úkol vypsat text získaný z textového pole na stránku. Popisovaný kód je zobrazen níže:

```
<h2 class="sub-header">1. Ne-persistentni XSS</h2>
 <div class="row">
 <div class="col-lg-5">
 <div class="input-group">

 <form method="post" action="">
 <input type="text" name="data" class="form-control"
 placeholder="Zde␣vlozte␣hledany␣vyraz"
 value="<?=␣isset($_POST['data'])␣?
htmlspecialchars($_POST['data'])␣:␣''␣?>"/>

<span class="input-group-btn">
<input class="btn␣btn-default" type="submit" name="submit"
 method="POST" value="Vyhledat" />
</span>
</form>

<?php
if(isset($_POST['submit'])) {
//navraceni vlozeneho vysledku na stranku
echo 'Pro␣hledany␣vyraz:␣<b>', $_POST['data'],
'</b>␣nebyl␣nalezen␣zadny␣vysledek.';
}
?>
</div>
</div>
</div>
```

V další části je PHP kód pro testování persistentního XSS útoku. V první části skriptu je zjišťováno, zda je soubor pro ukládání komentářů **lab1_komentar.txt** prázdný. Pokud ano, tak je uložena hláška `<b>Není vložen žádný komentář<b>` do proměnné **buff**. Ta je následně zobrazena na stránce. V případě, že není soubor prázdný, tak je obsah souboru nahrán do proměnné **buff** a zobrazen.

Vkládání komentářů je realizováno pomocí dvou vstupních formulářů - `field1` pro jméno a `field2` pro komentář a následném spuštění skriptu **lab1_save.php**.

Skript **lab1_save.php** nejprve vymaže obsah souboru **lab1_komentar.txt** a poté zapíše získané data do souboru. Následuje přesměrování na stránku **lab1.php**. Popisovaný skript je zobrazený níže:

```
//"vymazani"" souboru komentar
file_put_contents("lab1_komentar.txt", "");
if(isset($_POST['field1']) && isset($_POST['field2'])){
//naformatovani dat a ulozeni do promenne data
 $data = '<p><b>Jmeno: </b>' . $_POST['field1'] .
 '</p><p><b>Komentar: </b>' . $_POST['field2'] . '</p>';
 //zapsani dat promenne do souboru
 $ret = file_put_contents('lab1_komentar.txt',
 $data, FILE_APPEND | LOCK_EX);
    if($ret === false) {
        die('Chyba');
    }
    else {
        echo "$ret bytu zapsano do souboru";
    }
}
else {
   die('zadna data k odeslani');
}
//presmerovani na stranku lab1.php
header("Location:lab1.php");
```

V poslední části souboru **lab1.php** se nachází jednoduchý Javascript kód pro simulaci lokálních (DOM-based) XSS útoků. Jelikož jsou dnes moderní webové prohlížeče zabezpečené proti tomuto typu útoku, je těžké ho prakticky simulovat, a proto bylo zvoleno následující řešení.

Níže uvedený kód, má za úkol porovnat hodnotu parametru `jmeno` v URI. Pokud je hodnota parametru: `alert("XSS")` či `alert(document.cookie)`, je zobrazeno vyskakovací okno s uvedeným obsahem. V opačném případě je hodnota proměnné vypsána na stránce. Popisovaný kód:

```
<script>
var pos = window.location.href.substr(window.location.href.
lastIndexOf("?")+7);

if( pos.indexOf('alert(%27XSS%27)') >= 0){
alert("XSS");
}
if( pos.indexOf('alert(document.cookie)') >= 0){
alert(document.cookie);
}
else{
document.write("Vitej " + pos);
}
</script>
```

Druhá Laboratorní úloha **LAB2 - SQL injection** se skládá ze 2 souborů: **lab2.php** a **lab2start.php**. V momentě, kdy uživatel klikne na odkaz druhé laboratorní úlohy LAB2, dojde ke spuštění skriptu **lab2start.php**, který opět slouží pro inicializaci úlohy.

V tomto skriptu dojde k připojení databáze **test**. Dále skript ověří, zda existuje tabulka `sqlinjection`, pokud ano, dojde k jejímu vymazání.

V následujícím kroku dojde k opětovnému vytvoření tabulky sqlinjection s následující strukturou:

Tab. 7.8: Struktura tabulky sqlinjection

Název	**Typ**	**Další**
zamKod	varchar(30)	PRIMARY_KEY
uzivJmeno	varchar(30)	
celeJmeno	varchar(30)	
heslo	varchar(50)	
plat	int(6)	

Hesla opět nejsou uložena jako otisk z důvodu realizace SQL útoku. V reálném prostředí by opět měly být uložené otisky hesel.

V následující části dojde k naplnění tabulky `sqlinjection` ukázkovými daty, viz tab. 7.9. Na konci skriptu dojde k přesměrování na soubor **lab2.php**.

Tab. 7.9: Ukázková data pro LAB2 - SQL injection

zamKod	uzivJmeno	celeJmeno	heslo	plat
A3hHXn6532	kotrbajiri	Jiří Kotrba	kolotoc98	9680
y2WtD2TqP1	pavelkajan	Pavel Pelikán	skok56	15545
nhG1tY0A79	novotnykarel	Karel Novotný	134alarm	14320
60Ji6kY2MQ	novakovadana	Dana Nováková	jahody33	19530
Lxr2rDZNr3	kubovypavel	Pavel Kubový	11audi	24420

Skript **lab2.php** obsahuje HTML stránku se samotnou úlohou. V první části (1. Neošetřený vstupní formulář) se nachází vstupní formulář. Po vyplnění formuláře a potvrzení je spuštěn PHP skript, který opět připojí databázi **test** a v tabulce `sqlinjection` vyhledá vložený kód. V případě, že není kód nalezen, dojde k navrácení hlášky: `<h4>Byl zadán špatný zaměstnanecký kód!</h4>`, v opačném případě dojde k vypsání proměnné `celeJmeno` a `plat` dle nalezeného `zamKod`, který byl zadán. Popisovaný kód:

```
if(isset($_POST['submit'])){
//pripojeni databaze
mysql_connect('localhost', 'root', 'ubuntu');
mysql_select_db("test");

$id = $_POST['data'];
//dotaz na vyhledani radku s dle promenne id ve sloupci zamKod
$query = "SELECT␣*␣FROM
sqlinjection␣WHERE␣zamKod␣='" . $id . "'";
$q = mysql_query($query);
//pokud nebyl nalezen zadny radek
if (mysql_num_rows($q) == 0) {
printf("<h4>Byl␣zadan␣spatny␣zamestnanecky␣kod!</h4>");
}
else
{
//vypsani nalezenych radku
while ($row = mysql_fetch_array($q)) {
printf("<h4>Vase␣jmeno:␣<b>␣%s␣</b>,
Vyplatni␣paska␣k␣tomuto␣mesici:<b>␣%s
Kc</b>␣<br></h4>", $row["celeJmeno"], $row["plat"]);
}}}
```

Následuje druhá část (2. Neošetřený přihlašovací formulář). V tomto případě se jedná o klasický přihlašovací formulář složený z uživatelského jména a hesla.

V momentě vyplnění uživatelského jména, hesla a potvrzení dojde opět k připojení databáze **test**. Následně je vytvořen mysql dotaz k nalezení řádku se zadaným uživatelským jménem **username** a heslem **password**. V případě nalezení dojde k vypsání hlášky o úspěšném přihlášení uživatele a vypsání přihlášeného uživatelského jména z databáze. V opačném případě je vypsána hláška: `<h4>Špatné uživatelské jméno nebo heslo!</h4>`. Níže je zobrazen popisovaný kód:

```
if(isset($_POST['submit2'])){
if (empty($_POST['username']) || empty($_POST['password'])){
$error = "Vyplnte uzivatelske jmeno a heslo";
}
else
{
$username=$_POST['username'];
$password=$_POST['password'];
//pripojeni databaze
mysql_connect('localhost', 'root', 'ubuntu');
mysql_select_db("test");
//vyhledani radku dle uzivatelskeho jmena a hesla
$query2 = "SELECT * FROM sqlinjection WHERE uzivJmeno ='"
.$username. "' AND heslo='" .$password. "'";
$q2 = mysql_query($query2);
//pokud nebyl uzivatel nalezen
if (mysql_num_rows($q2) == 0) {
printf("<h4>Spatne uzivatelske jmeno nebo heslo!</h4>");
}
else //pokud byl uzivatel nalezen
{   while ($row2 = mysql_fetch_array($q2)) {
printf("<h4>Uzivatel <b> %s </b> byl uspesne prihlasen.</h4>"
, $row2["uzivJmeno"]);
}
}
}}
```

Testovací stránka pro SSLstrip útoky

Ve třetí části laboratorní úlohy je realizován útok SSLstrip. K tomuto útoku byla vytvořena ukázková stránka pro ssl strip. Stránka se nachází na serveru v adresáři **/var/www/html/sslstrip**.

Testovací stránka pro DOS útoky

Ve čtvrté části laboratorní úlohy jsou realizovány útoky typu DOS. K těmto účelům byla vytvořena webová stránka, kterou lze nalézt v adresáři **/var/www/html/dos**.

Ve stránce je využita fotografie stažena z webu pixbay.com. Stažený obrázek je dostupný z URL: pixbay.com/cs/service... Obrázek je šířený pod licencí **Creative Commons Deed CC0**, je tedy možné ho volně použít i ke komerčním účelům bez nutnosti uvádět zdroj.

8 ZÁVĚR

V diplomové práci byla zachycena problematika počítačových útoků. Bylo popsáno shromažďování informací o cíli, u kterého proběhlo rozebrání jednotlivých metod, jako je zjišťování veřejně dostupných informací, zjišťování dostupnosti, skenování portů, detekce OS a odhalení zranitelností. Pro realizaci shromažďování informací o cíli je ideální využití programu nmap, jelikož je schopný realizovat drtivou většinu z uvedených metod.

Část zabývající se počítačovými útoky byla rozdělena dle bezpečnostních vlastností systému, na které jsou útoky zacíleny. V případě útoků na důvěrnost byly popsány metody pro odposlouchávání, kterými jsou MAC záplava a odcizení portu. U útoků na autentičnost byly popsány čtyři známé a používané metody, těmi jsou ARP spoofing, Přetečení na zásobníku, Cross-site scripting a SQL injection. Pro ARP spoofing lze využít utilitu arpspoof, případně program Ettercap. Pro testování útoku přetečení na zásobníku je možné využít framework Metasploit, který obsahuje množství připravených exploitů. Útoky na dostupnost popisují několik metod DoS útoků jako jsou například TCP záplavy, UDP záplavy, DNS zesilující útok či Peer-to-peer útok. Pro realizaci většiny útoků DoS je možné využít programu hping3.

Na základě získaných znalostí lze říci, že je možné všechny popisované útoky realizovat za pomoci virtualizace. Limitujícím faktorem však zůstává hardwarový výkon počítače.

V praktické části byla vybrána technologie pro testování útoků v rámci virtuálního prostředí. Byl zvolen virtualizační nástroj VMware player a jako nástroj pro penetrační testování Kali Linux. Kromě funkce snapshoot obsahuje VMware player všechny potřebné vlasnosti pro vytvoření požadované úlohy. Tento nedostatek byl při tvorbě virtuálních počítačů vyřešen za pomoci trial verze VMware Workstation. Distribuce Kali Linux byla zvolena, jelikož obsahuje všechny nástroje pro realizaci popisovaných útoků.

Pomocí vybraných technologií bylo vytvořeno virtuální testovací prostředí pro laboratorní úlohy. Proběhla instalace, konfigurace třech virtuálních počítačů (klient, útočník, server) a vytvoření webové aplikace na virtuálním počítači serveru. V testovacím prostředí byly realizovány vybrané útoky.

Na základě provedených útoků byl vytvořen soubor čtyř laboratorních úloh. V první úloze je provedena základní analýza síťové komunikace využitím programu Wireshark. Ve druhé úloze je realizován síťový útok ARP spoofing, DNS spoofing a zprovozněna phishingová stránka. Třetí část se zabývá složitější technikou útoku SSL striping a útoky na webové stránky Cross-Site scripting a SQL injection. Ve čtvrté úloze je otestováno několik DoS útoků (ICMP flood, TCP flood, UDP flood, TCP reset) a útok na operační systém s využitím frameworku Metasploit. Všechny po-

pisované útoky byly úspěšně provedeny a jsou realizovatelné pomoci připravených virtuálních počítačů a vytvořených laboratorních návodů.

Ke všem laboratorním úlohám byly vytvořeny komentované videoukázky popisující provedení útoku, dále byla vytvořena dokumentace pro vyučujícího.

Všechna nastavení virtuálních počítačů, přístupová hesla a popis webové aplikace jsou v práci zdokumentovány. V budoucnu je tedy možné na základě dokumentace rozšířit soubor úloh o další úlohy, případně současné úlohy modifikovat.

LITERATURA

[1] Česká Republika. TRESTNÉ ČINY PROTI MAJETKU: Neoprávněný přístup k počítačovému systému a nosiči informací. In: *č. 40/2009 Sb.* 2009. Dostupné z: <http://www.zakonyprolidi.cz/cs/2009-40>

[2] *Hacking: manuál hackera.* 1. vyd. Praha: Grada 2008 399 s. ISBN 978-80-247-1346-5.

[3] JEŘÁBEK, Jan. *Pokročilé komunikační techniky.* Brno: Vysoké učení technické v Brně 2012. ISBN 978-80-214-4636-6.

[4] PETERKA, Jiří. *EArchiv.cz: Co je čím ... v počítačových sítích* [online]. 1991 [cit. 2014-11-22]. Dostupné z: <http://www.earchiv.cz/i_coje.php3>

[5] STUART MCCLURE, Joel Scambray. *Hacking exposed 7 network security secrets.* New York: McGraw-Hill, 2012. ISBN 978-007-1780-292.

[6] TCP handshake krok za krokem. ODVÁRKA, Petr. *Svět sítí* [online]. 2000 [cit. 2014-11-23]. Dostupné z: <http://www.svetsiti.cz/clanek.asp?cid=TCP-handshake-krok-za-krokem-3122000>

[7] Skenování portů: techniky. HALLER, Marin. *Lupa.cz* [online]. 2006 [cit. 2014-11-23]. Dostupné z: <http://www.lupa.cz/clanky/skenovani-portu-techniky/>

[8] UHLMANN, Stephan. ARP cache poisoning / ARP spoofing. *Su2* [online]. 2003 [cit. 2014-11-23]. Dostupné z: <http://su2.info/doc/arpspoof.php>

[9] Seriál Útoky typu DoS. HALLER, Martin. *Lupa.cz* [online]. 2006 [cit. 2014-11-23]. Dostupné z: <http://www.lupa.cz/serialy/utoky-typu-dos/>

[10] DEFRAWY, Karim El, Minas GJOKA a Athina MARKOPOULOU. *BotTorrent: misusing BitTorrent to launch DDoS attacks.* 2007. Dostupné z: <http://www.ece.uci.edu/~athina/PAPERS/BotTorrent.pdf>

[11] Buffer Overflow. *OWASP* [online]. 2014 [cit. 2014-11-23]. Dostupné z: <https://www.owasp.org/index.php/Buffer_Overflow>

[12] Pokročilé techniky XSS. KÜMMEL, Roman. *Soom.cz* [online]. 2008 [cit. 2014-11-24]. Dostupné z: <http://www.soom.cz/clanky/485--Pokrocile-techniky-XSS>

[13] SQL Injection a zabezpečení. LEHOCKÝ, Zdeněk. *Programujte.com* [online]. 2007 [cit. 2014-11-24]. Dostupné z: <http://programujte.com/clanek/2007041802-sql-injection-a-zabezpeceni/>

[14] W3schools.com. *SQL Injection* [online]. [cit. 2014-11-24]. Dostupné z: <http://www.w3schools.com/sql/sql_injection.asp>

[15] Google Nexus 7 Pwn Pad - Kali Linux. *Soom.cz* [online]. 2014 [cit. 2014-12-04]. Dostupné z: <http://www.soom.cz/clanky/1142--Google-Nexus-7-Pwn-Pad-Kali-Linux>

[16] Odposloucháváme data na přepínaném Ethernetu. HALLER, Martin. *Lupa.cz* [online]. 2006 [cit. 2014-12-06]. Dostupné z: <http://www.lupa.cz/clanky/odposlouchavame-data-na-prepinanem-ethernetu-3/>

[17] *Getting Started with VMware Player.* 2014 [cit. 2014-12-06]. Dostupné z: <http://www.vmware.com/pdf/desktop/vmware_player70.pdf>

[18] Configuring Host-Only Networking. *VMware Workstation 9 Documentation Center* [online]. [cit. 2015-05-18]. Dostupné z: <https://pubs.vmware.com/workstation-9/index.jsp?topic=%2Fcom.vmware.ws.using.doc%2FGUID-93BDF7F1-D2E4-42CE-80EA-4E305337D2FC.html>

[19] VMware Player Pro FAQs. *Vmware* [online]. 2015 [cit. 2015-05-18]. Dostupné z: <http://www.vmware.com/products/player/faqs.html>

[20] Alert: vsftpd download backdoored. *SECURITY HACKING EVERYTHING, BY CHRIS EVANS* [online]. 2014 [cit. 2015-05-18]. Dostupné z: <http://scarybeastsecurity.blogspot.cz/2011/07/alert-vsftpd-download-backdoored.html>

[21] How To Configure BIND as a Private Network DNS Server on Ubuntu 14.04. *Digitalocean.com* [online]. 2014 [cit. 2015-05-18]. Dostupné z: <https://www.digitalocean.com/community/tutorials/how-to-configure-bind-as-a-private-network-dns-server-on-ubuntu-14-04 >

[22] How To Create a SSL Certificate on Apache for Ubuntu 14.04. *Digitalocean.com* [online]. 2014 [cit. 2015-05-19]. Dostupné z: <https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-14-04 >

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AD	Active Directory
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
CAM	Content Addressable Memory
CRC	Cyclic redundancy check
CSS	Cascading Style Sheets
DDoS	Distributed denial of Service
DNS	Domain Name System
DOM	Document Object Model
DoS	Denial of Service
EBCDIC	Extended Binary Coded Decimal Interchange Code
EBP	Extended base pointer
EDNS	Extended domain Name System
ESP	Extended stack pointer
FTP	File Transfer Protocol
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IMAP	Internet Message Access Protocol
IP	Internet Protocol
ISO	International Organization for Standardization
LAN	Local Area Network

LDAP	Lightweight Directory Access Protocol
LIFO	last in, first out
MAC	Media Access Control
MITM	Man in the middle
NASL	Nessus Attack Scripting Language
OUI	Organizationally unique identifier
OWASP	Open Web Application Security Project
P2P	Peer-to-peer
PHP	PHP: Hypertext Preprocessor
RAM	Random Access Memory
RDP	Remote Desktop Protocol
SHA	Secure Hash Algorithm
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TTL	Time to live
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WWW	World Wide Web
XSS	Cross-site scripting

SEZNAM PŘÍLOH

A TEXT LABORATORNÍ ÚLOHY - ČÁST 1.

LABORATORNÍ ÚLOHA POČÍTAČOVÝCH ÚTOKŮ ČÁST 1.

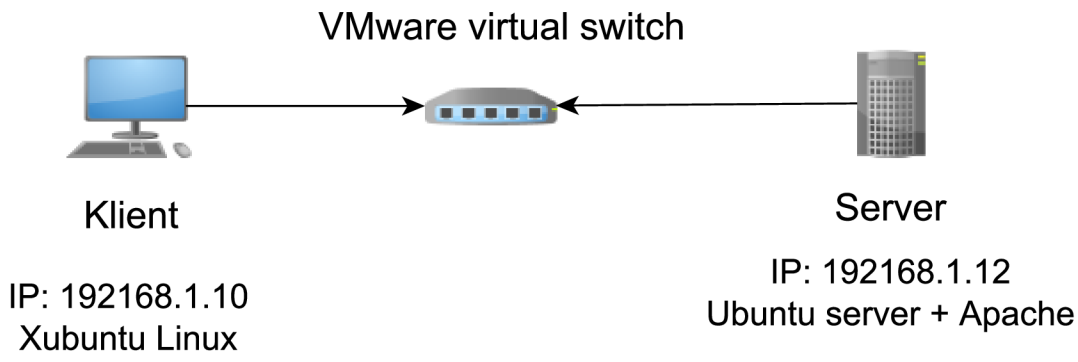
(Analýza síťové komunikace pomocí programu Wireshark)

Brno 2015

1 TEORETICKÝ ÚVOD

V rámci první laboratorní úlohy proběhne analýza síťové komunikace mezi klientem a serverem. Bude vysvětlena základní struktura HTTP protokolu, jeho metody i jeho slabiny. Následně proběhne analýza pomocí síťového analyzátoru Wireshark na počítači klienta. Laboratorní úloha má níže uvedenou topologii 1.1.

1.1 Topologie sítě laboratorní úlohy



Obr. 1.1: Topologie sítě laboratorní úlohy

1.2 Služba WWW a protokol HTTP

Služba WWW (World-Wide Web) se skládá ze třech základních technologií, kterými jsou HTML, URL a HTTP. HTML (HyperText Markup Language) je značkovací jazyk používaný pro tvorbu webových stránek, URL (Unique Resource Locator) slouží pro jednoznačné určení zdroje, umožňuje zapsat přesné umístění souboru na internetu, případně intranetu.

Pomocí protokolu HTTP (Hypertext Transfer Protocol) se k serveru odesílá URL stránky, kterou uživatel požaduje, server žádost zpracuje a opět pomocí protokolu HTTP odesílá uživateli stránku HTML. Protokol tedy pracuje na principu klient – server (**dotazy** od **klienta** a **odpovědi** od **serveru**). Je důležité říci, že HTTP je bezstavový protokol, to znamená, že mezi jednotlivými dotazy neexistuje žádná spojitost. První verze tohoto protokolu vznikla ve středisku CERN při vývoji systému WWW. V současné době existují již čtyři verze tohoto protokolu: HTTP 0.9, 1.0, 1.1 a poslední vydaná verze 2.0 vycházející z protokolu SPDY „Speedy“, který byl vytvořen společností Google.

V případě, že chce klient komunikovat se serverem, musí vytvořit TCP spojení. Vytvoření TCP spojení má tři kroky (anglicky three way handshake). V prvním kroku klient vytvoří inicializační paket SYN, vygeneruje sekvenční číslo SEQ (např. 350) a odešle serveru. Poté co dorazí SYN paket na server, server vygeneruje své vlastní číslo SEQ (např. 500), přijaté SEQ (350) od klienta inkrementuje o 1 a posílá zpět jako ACK (350 + 1). V posledním kroku klient získá toto potvrzení od serveru, inkrementuje SEQ zaslané serverem (500 + 1) a odesílá potvrzení synchronizace ACK (501) a SEQ (351). Tímto je TCP spojení vytvořeno.

V dalším kroku může přistoupit klient k vygenerování HTTP dotazu, dotaz má následující strukturu:

```
metoda cesta  verze_protokolu
Host: adresa_hosta
hlavičky
prázdná_řádka
data
```

Metoda

Protokol HTTP obsahuje několik metod. **Metoda** určuje druh dotazu, který je odeslán na server. Nejpoužívanější metoda je **GET**, pomocí této metody si klient vyžádá stránku specifikovanou dle **cesty** (URL adresa). Další velmi používanou metodou je **POST**, která se využívá pro odeslání dat na server (odeslání vyplněných formulářů, případně hesel). Metoda **HEAD** umožňuje klientovi získat hlavičku odpovědi serveru bez nutnosti serveru odesílat data, využívá se především ke zjištění, zda na stránce proběhla změna. Existují také metody, které nejsou v takové míře používány, sem patří metody **PUT** a **DELETE**, pomocí kterých je možné nahrát na server data, případně smazat, pokud má klient dostatečná oprávnění (ve většině případů je však dána přednost protokolu FTP). Další metody jsou **TRACE**, **CONNECT**, **OPTIONS**, tyto metody slouží primárně k analýze a nastavení způsobu spojení. Většinou jsou z hlediska bezpečnosti zakázané.

Cesta

Jedná se o URL adresu zdroje, kterou chce klient např. při použití metody GET získat.

Verze protokolu

Určuje verzi protokolu, kterou klient používá, v případě HTTP verze 1.1 bude obsahovat `HTTP/1.1`

Host

Uvádí doménové jméno serveru, ze kterého klient požaduje data. Toto pole

je od verze HTTP 1.1 povinné kvůli vzniku virtuálních serverů, jelikož na jednom fyzickém serveru může být spuštěno několik webových služeb.

Hlavičky

Hlavičky slouží na přenos atributů dokumentu, využívají standardu MIME (Multipurpose Internet Mail Extensions) pro přenos dokumentů elektronickou poštou. Hlavičky může umísťovat jak klient při vytváření dotazů, tak i server při odesílání odpovědí. Níže jsou uvedeny některé z nejčastěji používaných hlaviček:

Content-Type Tato hlavička určuje, o jaký typ přenášených dat se jedná. Například pro HTML stránky je typ `text/html`, pro text `text/plain`. Dále označení typů obrázků jako `image/gif`, `image/jpeg` či `image/png`.

If-Modified-Since V případě využití této hlavičky server odešle data pouze v případě, že byl dokument, který je dotazován, změněn po uvedeném datu v hlavičce.

User-Agent, Server Klient pomocí této hlavičky odesílá informace o typu prohlížeče. Server odesílá svojí identifikaci. Tato hlavička se hojně využívá při optimalizaci webových stránek.

Content-Type Specifikuje velikost těla „dat“ zprávy.

Accept-Charset Klient informuje server, jakou znakovou sadu podporuje.

Prázdná řádka

Odděluje hlavičku dotazu či odpovědi od zasílaných dat.

Příklad dotazu vygenerovaného klientem:

```
/*Klient žádá pomocí metody GET o soubor typu index.php*/
GET index.php HTTP/1.1
/*Žádá doménu www.test-domena.cz*/
Host: www.test-domena.cz
/*Používá prohlížeč Mozilla Firefox v28 pod operačním systémem Linux*/
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:28.0) Firefox
/*Klient informuje server o typech dat, které podporuje*/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9
/*Jazykové mutace, které akceptuje*/
Accept-Language: cz, sk en;q=0.9, de;=0.5
/*Kompresní metody, které podporuje*/
Accept-Encoding: gzip, deflate
/*Žádá server, aby neukončoval TCP spojení*/
Connection: keep-alive
```

U řádku obsahujícího například jazykové mutace je možné si všimnout nastavených preferencí pro jednotlivé jazykové mutace. Prohlížeč může nastavovat váhy z inter-

valu 0 až 1.0. Váha varianty je zadána parametrem `q`. Čím větší je hodnota váhy, tím víc je hodnota preferována. Pokud váha není uvedena, je uvažována jako váha 1.

Příklad odpovědi serveru na předchozí dotaz:

```
/*Server odpovídá stavovým kódem 200 (požadavek úspěšně zapracován)*/
HTTP/1.1 200 OK
/*Odesílá typ HTTP serveru a jeho verzi*/
Server: Apache/2.4.7 (Ubuntu)
/*Odesílá verzi PHP serveru*/
X-Powered-By: PHP/5.5.9-1ubuntu4.7
/*Server použil na odeslaná data kompresní metodu gzip*/
Content-Encoding: gzip
/*Délka dat za hlavičkou*/
Content-Length: 589
/*Server neuzavírá TCP spojení*/
Connection: Keep-Alive
/*Odeslaná data jsou ve formátu HTML*/
Content-Type: text/html
/*Nyní následuje prázdná řádka po níž začínají data*/

..........T.n.0... .p.\Fh....
```

Protokol HTTP definuje několik stavových kódů. Server generuje stavové kódy a tím informuje klienta o průběhu zpracování dotazů. Na níže uvedené tabulce jsou seřazené jednotlivé kódy do kategorií.

Tab. 1.1: Stavové kódy a hlášení HTTP/1.1

Kategorie stavového kódu	**Číslo stavového kódu**	**Popis**
Informační	100 - 199	Zpráva definovaná konkrétní aplikací
Úspěch	200 - 299	Požadavek byl úspěšně zpracován
Přesměrování	300 - 399	Klient musí pro konečné zpracování požadavku vykonat určitou další činnost
Chyba klienta	400 - 499	Problém na straně klienta
Chyba serveru	500 - 599	Problém na straně serveru

Protokol HTTP od verze 1.1 podporuje takzvané udržované spojení. To znamená, že TCP spojení není ukončeno po zpracování jednoho požadavku. Klient může požadovat vytvoření trvalého spojení se serverem tím, že umístí do hlavičky **Connection** hodnotu `keep-alive`. Server odpoví hlavičkou **Keep-Alive** a klient tak může zaslat

další požadavek po stejném TCP spojení. Důvodem zavedení udržovaného spojení je hlavně úspora přenášených paketů, jelikož vytvoření TCP spojení je náročný proces.

S udržovaným spojením úzce souvisí i hlavička **Content-Lenght**. U protokolu HTTP 1.0 klient poznal konec dat tím, že server uzavřel spojení. U udržovaného spojení však klient nemá jinou možnost, jak rozpoznat konec dat, než z deklarované délky dat **Content-Lenght**.

V poslední části je důležité zmínit, že protokol HTTP neobsahuje kromě autentizační metody HTTP Digest žádné jiné mechanismy pro šifrování přenosu. Je tedy snadné přenášená data odposlechnout. K ošetření této slabiny se využívá nadstavby HTTPS, pomocí které je možné data zabezpečit využitím šifrování pomocí SSL (Secure Sockets Layer) nebo TLS (Transport Layer Security) protokolů, které budou rozebrány ve třetí části laboratorních úloh.

1.3 Síťová analýza a sniffing

Analýza sítě (často označovaná jako sniffing) je proces, při kterém dochází k zachytávání síťového provozu a následnému detailnímu rozboru. Síťový analyzátor dekóduje datové pakety do pro člověka srozumitelné podoby. Sniffer je program, který zkoumá přenášená data v síti. Síťovým analyzátorem může být speciální hardwarové zařízení nebo může jít o software nainstalovaný na běžném počítači.

Síťový analyzátor je nezanedbatelným pomocníkem při řešení problémů se sítí a její konfigurací. Využívají ho systémoví administrátoři, síťoví a bezpečnostní odborníci a mnoho dalších. V minulosti byly síťové analyzátory speciální a drahá hardwarová zařízení. Vlivem technologického pokroku vznikly i výše zmíněné softwarové analyzátory. Snadná dostupnost těchto nástrojů má svá pro i proti, zatímco systémoví administrátoři analyzátory využívají k řešení problémů a monitorování sítě, útočníci používají analyzátor k páchání škod.

Pokud je analyzátor (sniffer) využíván člověkem se špatným úmyslem, může představovat významnou hrozbu pro bezpečnost sítě. V této souvislosti je také často slovo sniffing dáváno do souvislosti s významem odposlech. Neoprávněné používání snifferu je považováno za pasivní formu útoku. Útočníci používají sniffery pro následující účely:

- zachytávání uživatelských jmen a hesel v podobě prostého textu,
- odhalení vzorců chování uživatelů sítě,
- kompromitace důvěrných informací,
- zachytávání a přehrávání telefonní konverzace prostřednictvím ptokolu VoIP,
- mapování síťové topologie,
- pasivní detekce operačního systému.

Je důležité připomenout, že pokud nejsou výše uvedené body prováděné pověřeným penetračním testerem, tak jsou považovány za nezákonné!

1.4 Seznámení s programem Wireshark

Wireshark je velmi rozšířený síťový analyzátor. Vychází ze svého předchůdce známého pod názvem Ethereal, který v roce 1977 vytvořil Gerald Combs. Jedná se o doposud nejlepší dostupný open-source program pro analýzu sítě, který je dostupný pro více jak 20 platforem (Windows, Linux, Apple OSX, FreeBSD a mnoho dalších). Obsahuje funkce srovnatelné s konkurenčními síťovými analyzátory, v některých ohledech konkurenci i přesahuje.

1.4.1 Vlastnosti Wiresharku

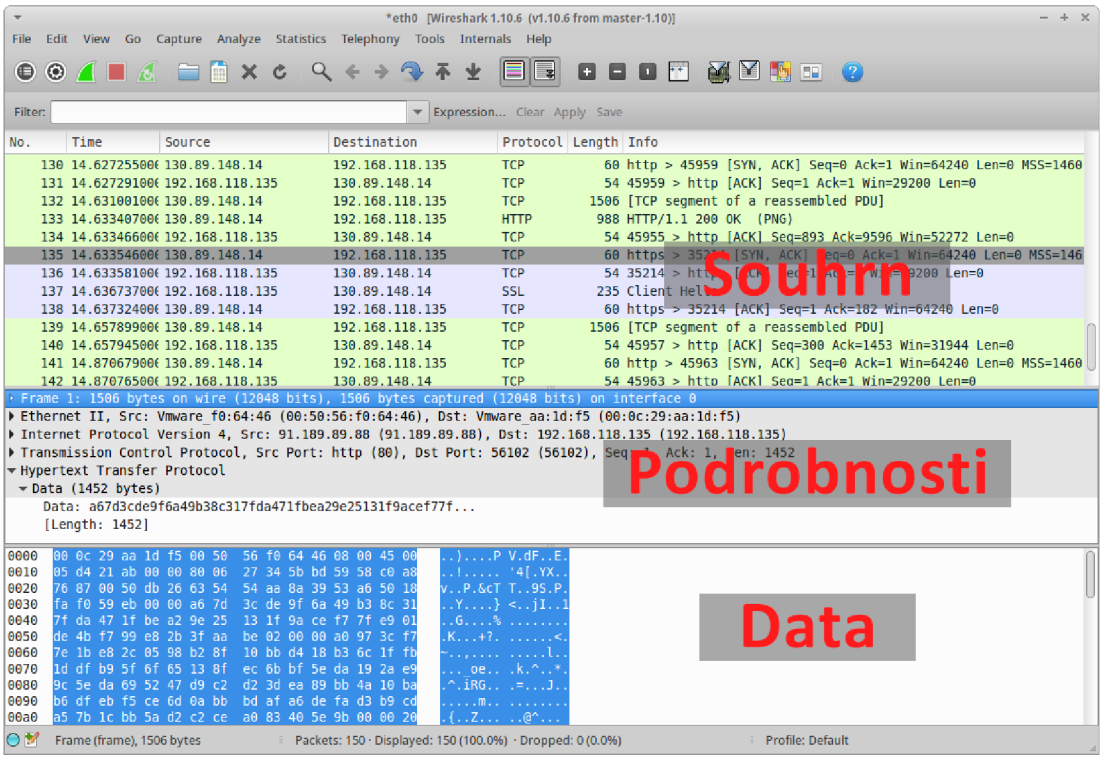
Pro správnou funkci Wiresharku je potřeba mít nainstalovaný ovladač. Za odesílání komunikace a příjem dat v síti je zodpovědná síťová karta, síťové aplikace využívají její metody, kterými mohou být například sokety pro navázání a udržení spojení. Využití síťových prostředků je realizováno skrze nižší vrstvy operačního systému, ten pak zprostředkovává přístup aplikacím. Není však vždy výhodou takto zprostředkovaného ovládání, data mohou být ovlivněna nebo upravena jinou aplikací. Dnes jsou proto společně s Wiresharkem známé a používané ovladače jako je **libpcap** nebo **WinPcap**. Tyto ovladače jsou schopné zachytit všechna surová data nacházející se v síti. Libcap je open-source ovladač určený převážně pro unix systémy, který kromě zachytávání dat může i za běhu některá data filtrovat. Protějškem pro platformu Windows je ovladač WinPcap.

Oba tyto ovladače jsou schopné zachytávat jak data určená pouze pro dané síťové rozhraní, na kterém pracují, tak i odchytávat data v celém segmentu sítě. Dle rozsahu zachytávání se dělí na promiskuitní režim a nepromiskuitní režim. **Promiskuitní režim** - síťová karta zaznamenává data z celého segmentu sítě. **Nepromiskuitní režim** - síťová karta zachytává data určená pouze pro její rozhraní.

1.4.2 Ovládání programu

Grafické rozhraní programu Wireshark se skládá ze tří panelů. Prvním je **souhrn**, na tomto panelu je zobrazen souhrn odchycených dat. Jsou zde informace, jako je čas odchycení paketu, zdrojová a cílová adresa a název protokolu nejvyšší vrstvy. Dalším panelem jsou **podrobnosti**, tento panel poskytuje detailní informace o všech vrstvách vybraného paketu. Informace jsou řazeny do stromové struktury. Posledním panelem jsou **data**, tento panel zobrazuje surová data v jejich hexadecimální

a textové podobě. Na níže uvedeném obrázku 1.2 je znázorněno rozdělení do tří panelů.



Obr. 1.2: Pohled na hlavní okno programu Wireshark

Důležitou funkcí ve Wiresharku jsou filtry. Ty umožňují jednodušší orientaci v zachycených datech. V případě, že je potřeba vyhledat konkrétní paket, je možné využít filtr zachycených paketů. Při zadávání filtrů se Wireshark drží syntaxe, která byla zavedena knihovnou libcap. Využít lze níže uvedené porovnávací operátory:

- **Rovnost:** eq, ==
- **Nerovnost:** not, ne, !=
- **Větší než:** gt, >
- **Menší než:** lt, <
- **Větší nebo rovno:** ge, >=
- **Menší nebo rovno:** le, <=

Pokud je například potřeba zobrazit pakety pouze se zdrojovou IP adresou 192.168.1.12, lze použít níže uvedený filtr:

```
ip.src == 192.168.1.12
```

Dále je možné řetězit jednotlivé výrazy pomocí jednoduché Booleovské logiky `AND` nebo `OR`. Příklad může být následující: Cílem je zobrazit pouze pakety směřující z IP adresy 192.168.1.12 s protokolem HTTP a metodou `GET`:

```
ip.src == 192.168.118.135 and http.request.method == "GET"
```

Využití filtrů je jistě velmi užitečným nástrojem při práci s Wiresharkem, který lze doporučit.

2 PRAKTICKÁ ČÁST

Praktická část je zaměřena na seznámení s programem Wireshark, jeho základním ovládáním, vytvářením filtrů a využitím pluginů pro hlubší analýzu zachycených dat. Bude provedena analýza protokolu HTTP na zachycených datech a nalezení citlivých informací.

2.1 Zachycení dat

V rámci praktické části budou všechny simulace probíhat ve virtuálním prostředí VMWare Playeru. Na ploše se nachází složka s názvem **Bezpečnost**, která obsahuje zástupce na potřebné virtuální počítače.

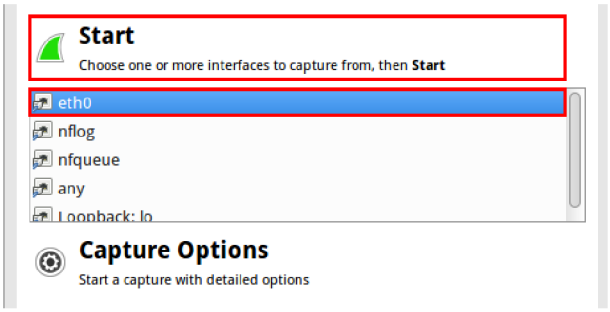
Nyní je potřeba spustit **Server - Ubuntu Server 14.04.2 LTS** a **Klient - Xubuntu 14.04**. Pro načtení operačních systémů je třeba vyčkat.

Jelikož nebude potřeba provádět žádné nastavení na serveru, může se ponechat minimalizovaný. Po načtení klientského počítače **Xubuntu** je případně možné ověřit vzájemnou komunikaci se serverem příkazem `ping 192.168.1.12`. Pomocí klávesové zkratky `windows + T` se vyvolá okno Terminálu.

Klient: V dalším kroku se přistoupí ke spuštění programu **Wireshark** a Prohlížeče **Mozilla Firefox**. Obě aplikace jsou k nalezení v **hlavní nabídce** v horním pravém rohu pod položkou **internet**, případně je možné využít vyhledávání.

Nyní se přistoupí k nastavení Wiresharku. Po spuštění se zobrazí základní okno Wiresharku, které je rozděleno do několika částí. V první části **Capture** jsou zobrazeny rozhraní, na kterých je možné spustit zachytávání. V části **files** je možné vybrat již uložená a zachycená data ze souboru, případně je zde možnost získat ukázková data z internetu. Část **Online** a **Capture Help** slouží jako nápověda.

Klient, program Wireshark: Pozornost je třeba věnovat části **Capture**, kde je nutné vybrat rozhraní **eth0** (pomocí tohoto rozhraní je virtuální počítač připojen do virtuální sítě) a spustit odchytávání pomocí zeleného tlačítka **start**, viz obr. 2.1.



Obr. 2.1: Výběr rozhraní ve Wiresharku

V tomto okamžiku běží zachytávání v reálném čase a je zobrazeno klasické rozhraní ze tří panelů, viz obr. 1.2. V dalším kroku bude generován síťový provoz mezi Klientem a Serverem.

Klient: Nyní je třeba se přepnout do prohlížeče Mozilla Firefox a otevřít stránku `www.test-domena.cz`, jedná se o webovou aplikaci běžící na Serveru (Zachytávání ve Wiresharku ponechat zapnuté).

Po načtení webové aplikace se zobrazí klasický přihlašovací formulář.

Klient: V tomto kroku je třeba vložit libovolné uživatelské jméno a heslo, následně se pokusit o přihlášení (Je vhodné zvolit uživatelské jméno a heslo tak, aby bylo snadno zapamatovatelné pro další analýzu, např. **Uživatelské jméno** `katerinaVob` a **heslo** `mojeTajneHeslo`).

Nyní byly vygenerovány data pro analýzu.

Klient: Následuje opět přepnutí do programu Wireshark, zde je nutné ukončit zachytávání červeným tlačítkem „Stop the running live capture“. Zachycená data je vhodné z hlediska zálohy uložit do souboru pomocí **File** a **Save As...**.

Zajímavou možností je zachycená data před uložením komprimovat zatržením volby **Compress with gzip**. Tímto je zachycení paketů kompletní.

2.2 Analýza dat

Zachycená data budou nyní podrobena analýze. Jelikož bude cílem prozkoumat pouze HTTP komunikaci využívající protokolu TCP je vhodné ostatní odchycené pakety odfiltrovat.

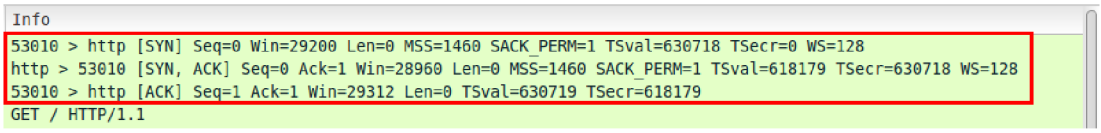
> **Klient:** V tomto případě je třeba použít jednoduchý filtr `tcp` dle obr. 2.2. Tímto je docíleno zobrazení pouze paketů využívajících TCP spojení.

Obr. 2.2: Aplikování filtru tcp

Se stejným výsledkem lze k filtrování přistoupit i opačně a určit protokoly, které nezobrazovat. Příkladem může být výraz:

```
not arp and not dns
```

Nyní by měly první tři odchycené pakety obsahovat sestavení TCP spojení obdobně jako na obrázku 2.3.



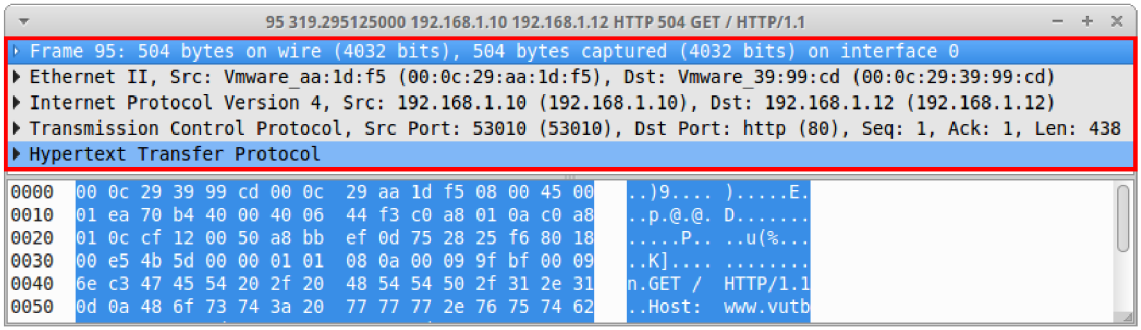
Obr. 2.3: Zachycená data po aplikování filtru

1. V prvním řádku zasílá klient serveru paket s příznakem **SYN** a s vygenerovanou hodnotou SEQ=0.
2. Na dalším řádku server odpovídá s příznaky **SYN** a **ACK**, server generuje své SEQ=0 a odesílá potvrzení na SEQ z prvního řádku + 1, ACK je se tedy rovná 1.
3. Ve třetím řádku klient potvrzuje spojení odesláním paketu s příznakem **ACK**, zasílá také své SEQ inkrementované o 1.

Čtvrtým zachyceným paketem je již dotaz od klienta s metodou **GET**.

> **Klient:** Dvojitým kliknutím na daný paket s metodou **GET** se dá panel s podrobnostmi zobrazit v novém okně, což usnadňuje orientaci.

Podrobnosti výsledného paketu budou obdobné jako na obr. 2.4.



Obr. 2.4: Okno s podrobnostmi o paketu

V podrobnostech jsou informace zobrazeny od nejnižší vrstvy (v tomto případě Ethernet II) po nejvyšší vrstvu (Hypertext Transfer Protocol). Nyní je možné nalézt samotný dotaz **HTTP** s metodou **GET** pro hlubší prozkoumání. Je nutné projít stromovou strukturou ve správné vrstvě k dotazu.

Obr. 2.5: Dotaz s metodou GET

Jak je vidět z obrázku 2.5, odchycená data odpovídají struktuře popsané v teoretickém úvodu 1.2. Za zmínku stojí znaky **\r\n**. Jedná se o kombinaci CRLF znaků, kde „\r“ je CR (Carriage return) znamená posunutí kurzoru na nový řádek a „\n“ je LF (Line feed), což znamená odřádkování. Je dobré si uvědomit, že pokud se vyskytují dva znaky **\r\n** za sebou, vzniká tím jeden prázdný řádek, a tím je oddělena hlavička od dat. Pomocí těchto znaků se detekuje i začátek dat.

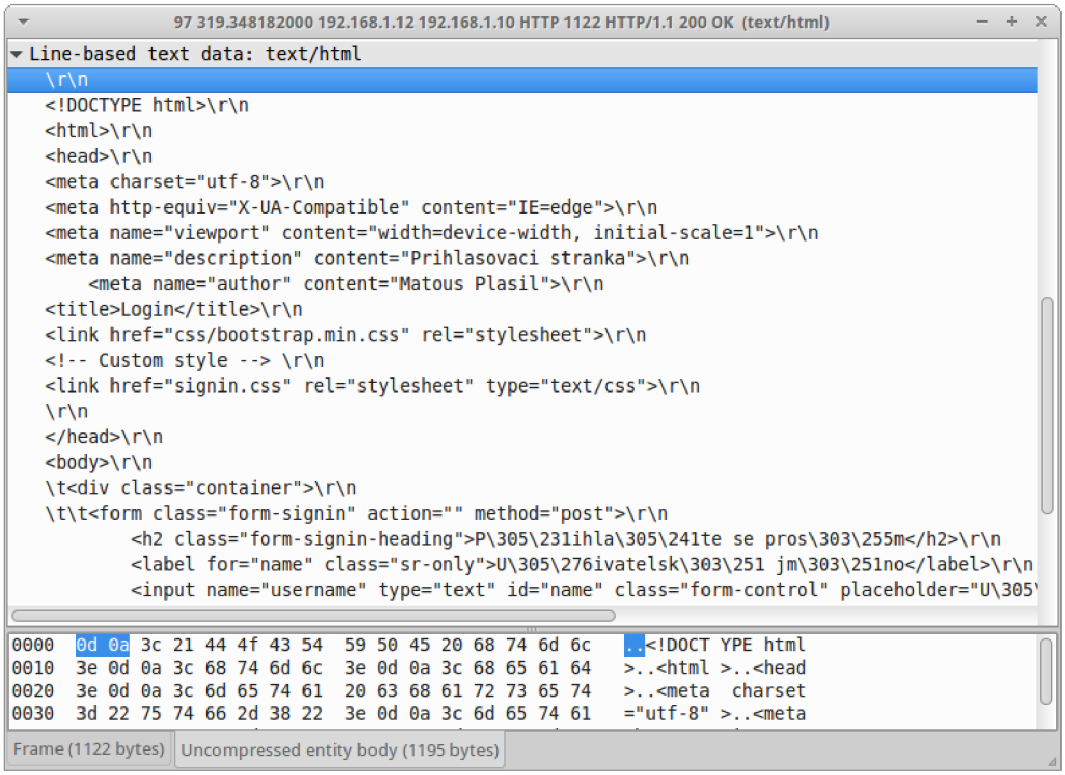
Protokol HTTP využívá pro svůj přenos TCP spojení, to znamená, že je spojení potvrzováno. Tyto potvrzení lze nalézt také v zachycených paketech - příznak **ACK**.

Po úspěšném odeslání **dotazu** následuje **odpověď** serveru. Nalezení odpovědi je možné pomocí vyhledání stavového kódu zaslaného serverem. Jelikož je nejčastější odpověď 200 (Požadavek úspěšně zpracován) je možné nalézt paket podle pole **Info**.

Klient: Dále je možné využít níže uvedený filtr:

```
http.response.code == 200
```

Po nalezení paketu je třeba zobrazit jeho detaily. Pomocí stromové struktury je možné zobrazit odeslaná data od serveru. A tím i celý jejich obsah, pomocí kterého je možné zpětně celou webovou stránku zrekonstruovat, viz 2.6.

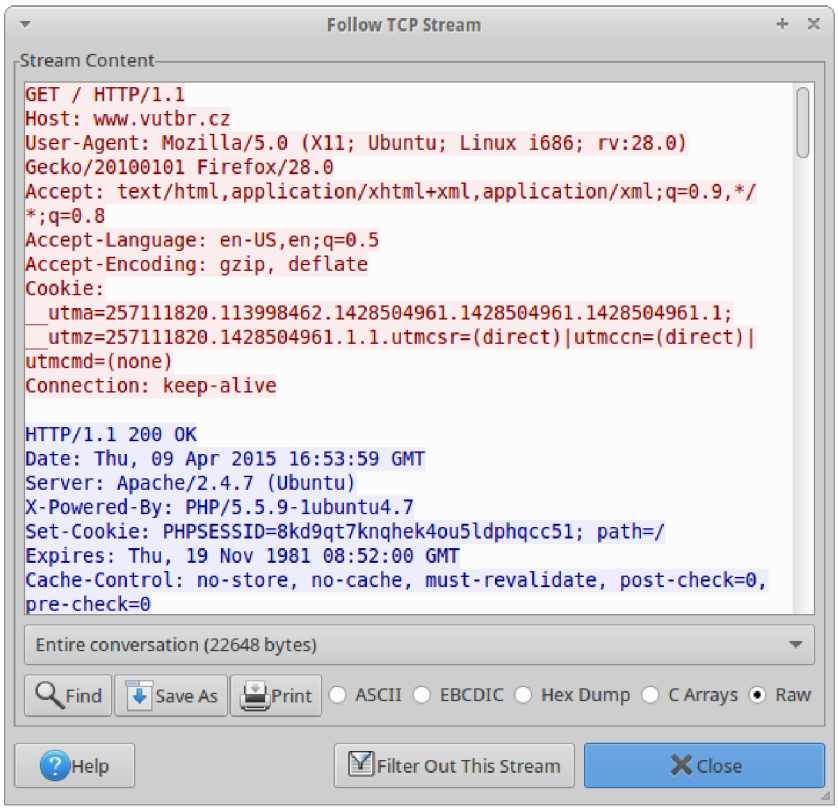


Obr. 2.6: HTML kód získaný v odpovědi od serveru

Pro jednodušší orientaci je zajímavé využít zobrazení přehledu komunikace v rámci TCP spojení, viz obr. 2.7.

Klient: Přehled se zobrazí pomocí kliknutí pravého tlačítka na vybraný paket a následnou volbou **Follow TCP Stream**.

Červeně označená data jsou ve směru od klienta, modře označená data jsou od serveru. V tomto zobrazení nejsou čitelná získaná data, jelikož nejsou dekomprimována (v případě zobrazení pomocí panelu **podrobnosti** byla data automaticky dekomprimována).



Obr. 2.7: Zobrazený přehled v rámci TCP spojení

2.3 Úkol

V rámci ověření pochopení problematiky je nyní úkolem najít **uživatelské jméno** a **heslo**, které bylo odesláno klientem na webovou aplikaci `www.test-domena.cz` s využitím vhodného filtru.

3 ZÁVĚR

Analýza síťové komunikace a protokolu HTTP je s programem Wireshark snadná. Snadné je také získání dat, a to jak hesel odeslaných od klienta, tak dat odeslaných serverem. Pokud není pro přenos hesla využita speciální metoda pro autentizaci HTTPS Digest, tak je odesílání hesla pomocí HTTP nebezpečné. Vhodnější variantou je využití nadstavby HTTPS (Hypertext Transfer Protocol Secure), kdy jsou data protokolu HTTP šifrována pomocí SSL (Secure Sockets Layer) nebo TLS (Transport Layer Security) protokolů.

3.1 Kontrolní otázky

1. Pomocí jaké hlavičky se při odesílání dotazu odesílá informace o verzi webového prohlížeče?
2. Jaký je rozdíl mezi promiskuitním a nepromiskuitním módem síťové karty?
3. V případě, že by byl útočník schopný odposlouchávat komunikaci realizovanou protokolem HTTP, získal by citlivé údaje a případná hesla?

LITERATURA

[1] OREBAUGH, Angela. *Wireshark a Ethereal: kompletní průvodce analýzou a diagnostikou sítí*. Vyd. 1. Brno: Computer Press, 2008, 444 s. ISBN 978-80-251-2048-4.

[2] KOSEK, Jiří. Základy protokolu HTTP. *Kosek.cz* [online]. 1999 [cit. 2015-04-11]. Dostupné z URL: <`http://www.kosek.cz/clanky/iweb/05.html`>.

[3] GRYGAREK, Petr. Základní vlastnosti protokolu HTTP. *VŠB Katedra informatiky* [online]. [cit. 2015-04-11]. Dostupné z URL: <`http://www.cs.vsb.cz/grygarek/kotasek/htp02.htm`>.

[4] LAMPA, Petr. Protokol HTTP 1.1. *Průvodce systémem WWW* [online]. 2002 [cit. 2015-04-11]. Dostupné z URL: <`http://www.fit.vutbr.cz/~lampa/WWW/http11.html.cs`>.

B DOKUMENTACE PRO VYUČUJÍCÍHO - LABORATORNÍ ÚLOHA - ČÁST 1.

B.1 Základní informace

V této úloze mají studenti získat základní přehled o mechanismech a metodách používaných při komunikaci pomocí protokolu HTTP. Dále získají teoretické znalosti o síťové analýze pomocí programu Wireshark.

V praktické části probíhá odchycení dat na počítači klienta a jejich následná analýza. Jsou popsány některé filtry využívané v síťovém analyzátoru Wireshark. Studenti mají k úloze spuštěn virtuální počítač klient a server. Všechna konfigurace však probíhá pouze na virtuálním počítači klienta.

Kontrolní úkol v závěru úlohy má otestovat nabyté znalosti o protokolu HTTP a schopnost práce s programem Wireshark.

B.2 Řešení úkolu

Studenti mají najít využitím vhodného filtru odeslané heslo v zachycených datech. Řešením je využití níže uvedeného filtru:

```
http.request.method == POST
```

A následné rozkliknutí detailu paketu, kde je pod položkou **Line-based text data: text/html** možné zjistit zaslané heslo.

B.3 Odpovědi na otázky

1. Informace o verzi webového prohlížeče se odesílá pomocí hlavičky **User-Agent:**.
2. U promiskuitního režimu síťová karta zaznamenává data z celého segmentu (tedy i data, které jí nejsou určená). Naopak u nepromiskuitního režimu jsou zachytávaná data určená pouze pro dané rozhraní.
3. Ano mohl. Data protokolem HTTP jsou přenášena nešifrovaně. Je tedy možné zobrazit jak informace odeslané serverem (webové stránky, obrázky, apod.), tak informace zaslané klientem (uživatelská jména, hesla, obsah formulářů).

C TEXT LABORATORNÍ ÚLOHY - ČÁST 2.

LABORATORNÍ ÚLOHA POČÍTAČOVÝCH ÚTOKŮ ČÁST 2.

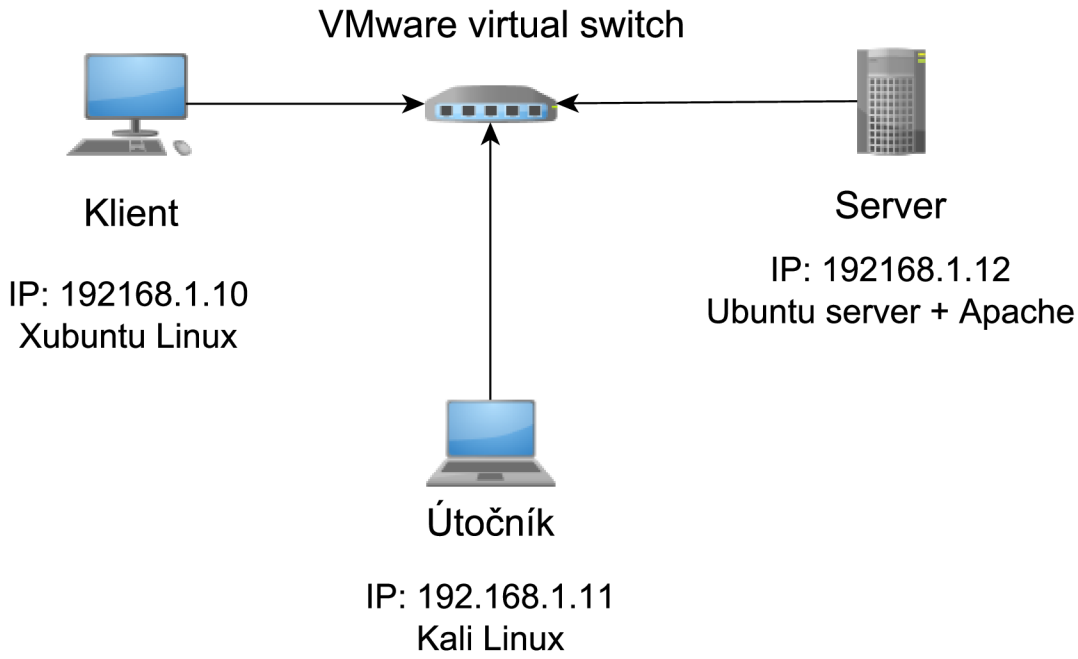
(Realizace síťového útoku s využitím operačního systému Kali Linux)

Brno 2015

1 TEORETICKÝ ÚVOD

V této laboratorní úloze bude vysvětlen princip ARP protokolu a DNS protokolu. Následně dojde k realizaci jednoduchého útoku MITM (Man in the middle) za pomoci techniky ARP spoofing. MITM je útok, kdy se útočník dostane mezi komunikační kanál server/klient. Poté dojde k realizaci phishingu za pomoci techniky DNS spoofing. Využívána bude níže uvedená topologie 1.1.

1.1 Topologie sítě laboratorní úlohy



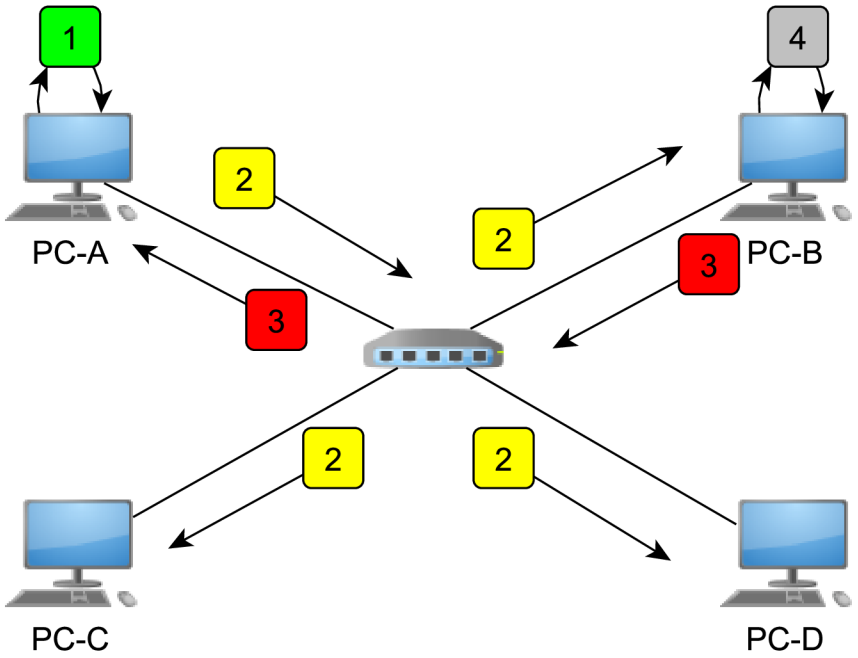
Obr. 1.1: Topologie sítě laboratorní úlohy

1.2 ARP protokol

Protokol ARP slouží k nalezení fyzické „MAC“ adresy na základě logické „IP“ adresy. Nalezení odpovídající fyzické adresy je **nezbytné** pro úspěšnou komunikaci.

ARP využívá pomocné tabulky dočasných záznamů **ARP cache**. Do těchto tabulek ukládá PC nebo síťový prvek IP adresy a jejich fyzické adresy. Tyto záznamy se naplní buď pomocí ARP protokolu (pokud potřebuje stanice získat fyzickou adresu jiné stanice za účelem odeslání paketu) nebo se dají zadat tyto informace manuálně.

Na níže níže uvedeném obrázku 1.2 je znázorněn proces komunikace při ARP požadavku.



Obr. 1.2: Schéma komunikace (ARP request)

1. PC-A chce odeslat data PC-B. V prvním kroku si PC-A prozkoumá svojí tabulku „ARP cache“, zda nenajde informaci odpovídající k hledané logické adrese.
2. V případě že PC-A nenalezne záznam o adrese PC-B, tak vyšle žádost „request“ ve formě broadcastu do sítě. Tuto žádost obdrží všechny PC v rámci broadcastové domény.
3. Až na PC-B všechny stanice žádost zahodí. PC-B odešle odpověď „response“ ve formě unicastu PC-A s vyplněným polem hledané fyzické adresy
4. PC-B si zároveň zkontroluje obsah své tabulky a případně jí doplní o informaci obdrženou v žádosti ARP od PC-A.

1.3 DNS protokol

1.3.1 Systém DNS

Úkolem DNS systému je převod doménových jmen např. „www.vutbr.cz“ na IP adresy „147.22.9.2.90“ a naopak. Důvodem celého systému je umožnit uživatelům

využívat místo IP adres doménová jména, jelikož jsou mnohem snadnější na zapamatování.

DNS systém tvoří stromovou hierarchii, která začíná DNS kořenem (root), kořen by měl být vyjádřen tečkou **.** , ve většině případů se však nepíše. Pod kořenem se nachází takzvané TLD domény (Top-Level Domain), jako je například **cz**, **com** nebo **org**. Pod těmito doménami jsou následně další zóny, tyto zóny vlastní buď organizace, nebo soukromé osoby, příklad zóny: **vutbr**. Tyto jednotlivé koncové zóny pak mohou obsahovat další subdomény, jako příkladem může být subdoména **kam**. Výsledná adresa může vypadat následovně: **www.kam.vutbr.cz.**. Nejvyšší úroveň se tedy nachází napravo (.), nejnižší nalevo (www), přičemž jednotlivé sekce jsou v hierarchii odděleny tečkou.

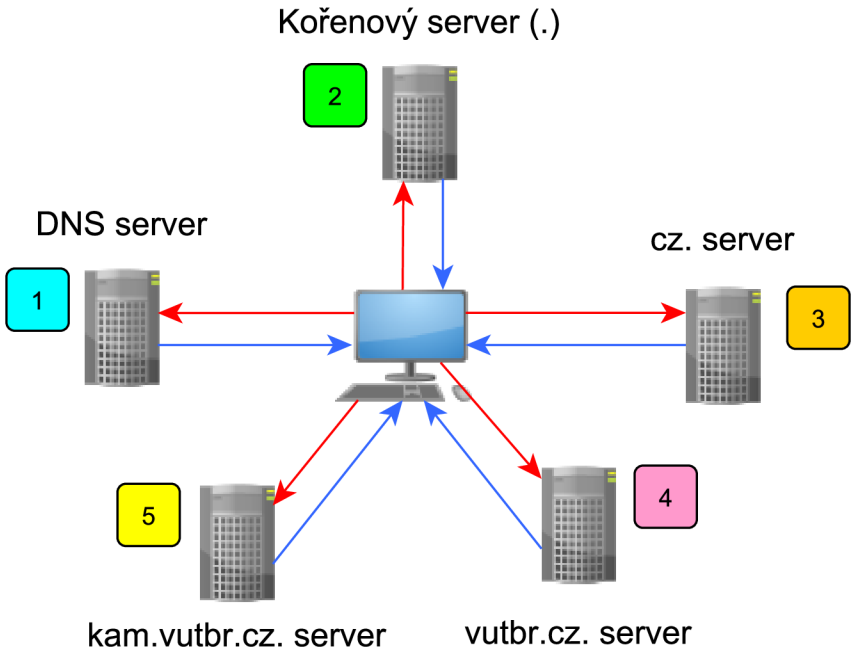
Každá zóna musí mít jeden primární (MASTER) a minimálně jeden sekundární (SLAVE) DNS server. Primární udržuje informace o dané zóně, sekundární DNS servery si pravidelně kopírují nové informace ze serveru primárního. Sekundární servery ověřují stav sériového čísla dané zóny, pokud je sériové číslo větší, je nutné aktualizovat údaje z primárního serveru.

1.3.2 Protokol DNS

Pomocí tohoto protokolu se uskutečňuje interakce mezi resolverem (klientem) a DNS serverem. Komunikace je tedy založena na principu klient/server. Protokol DNS spadá do aplikační vrstvy, přenos je realizován pomocí protokolu UDP nebo TCP na portu 53. U dotazů na překlad je přednost dávána protokolu UDP. V případě DNS existují dva základní typy dotazů:

- **Rekurzivní DNS dotaz** - DNS překladač se na klientské stanici dotazuje na překlad doménového jména např. www.vutbr.cz na IP adresu. Žádá ho o doručení konečného výsledku, tedy požadované IP adresy. V tomto případě jsou nejvíce namáhány prostředky DNS serveru.
- **Iterativní DNS dotaz** - Resolver žádá DNS server o zaslání odkazů (referrals) na další DNS servery, u kterých ví, že obsahují další potřebné informace. V tomto případě je proces vyhledávání realizován u resolveru, tedy klienta.

Níže je uvedený průběh iterativního DNS dotazu viz. obr 1.3.



Obr. 1.3: Iterativní DNS dotaz

1. Resolver zasílá dotaz a žádá o zaslání odkazů na další servery, které by budou schopné dodat potřebné informace. DNS server odpovídá zasláním odkazu na Kořenový server.
2. Resolver opět zasílá dotaz na kořenový server. Kořenový server odesílá odkaz na server určený pro danou TLD doménu.
3. Resolver nyní kontaktuje cz. server. Cz. Server odesílá odkaz na server s adresou vutbr.cz.
4. Resolver odesílá dotaz vutbr.cz. serveru. Server odpovídá opět odkazem na kam.vutbr.cz server.
5. Resolver kontaktuje kam.vutbr.cz., který mu sdělí finální adresu 147.22.9.2.20.

1.4 Kali Linux

Kali Linux je Linuxová distribuce společnosti Offensive-security, která vyšla v roce 2013 a je určena pro pokročilé penetrační testování a realizaci bezpečnostních auditů. Tato distribuce je následovníkem Linuxové distribuce Black Track 5 a byla vytvořena stejnou společností.

Minimální konfigurace pro běh Kali Linuxu na architektuře i386 či amd64 je 1 Ghz CPU, 8 GB HDD a 300MB RAM. Kromě i386 a amd64 je podporováno několik

různých ARM platforem, jako je rk3306/ss808, Raspberry Pi, ODROID U2/X2, Samsung Chromebook či Galaxy Note 10.1.

Kali Linux je postaven na základě OS Debian (verze 7.0 „Wheezly“) a je tedy kompatibilní s **.deb** balíčky. V základní verzi je podporováno více jak 300 nástrojů pro penetrační testování. Níže jsou jedny z nejznámějších uvedené:

Aircrack-ng
: Jedná se o nástroj pro testování zabezpečení Wi-Fi sítí. Nejčastěji se využívá k prolomení hesla do bezdrátové sítě zabezpečené pomocí WEP nebo WPA-PSK (Nadstavbou tohoto nástroje je program **Wifite**, což je velmi jednoduchý program pro získání hesla z Wi-Fi sítí používající WEP, a **Fern Wi-Fi Cracker**, který se na základě brute force slovníkového útoku snaží zjistit WPA klíč).

Dsniff
: Jedná se o sniffer sloužící pro zachytávání mnoha protokolů. Automaticky analyzuje aplikační protokoly a poté ukládá informačně zajímavé části, jako jsou vyplněné formuláře nebo hesla (součástí je i **Arpspoof**, který slouží k realizaci MITM skrze ARP protokol).

SSLstrip
: Program zachytává odchozí HTTP komunikaci oběti a přepisuje příchozí odkazy HTTPS na HTTP, tím nutí oběť k používání HTTP komunikace, se serverem však udržuje HTTPS spojení. Slouží k odchycení komunikace.

Ettercap
: Jedná se o LAN sniffer a zároveň velmi silný nástroj pro MITM útoky (plugin **remote_browser** dovoluje například sledovat WWW stránky na které oběť surfuje).

Metasploit
: Open source framework pro penetrační testování, obsahuje informace o známých slabinách, obsahuje kódy využitelné po vniknutí do zranitelného systému (payload). Možnost využít šifrovací techniky k odhalení payloadu pomocí IPS (Intrusion-prevention System).

2 PRAKTICKÁ ČÁST

V rámci praktické části bude realizován útok typu MITM (Man in the middle). Bude odposlechnuta komunikace mezi klientem a serverem, která bude následně analyzována obdobně jako v 1. části laboratorní úlohy.

V další části proběhne realizace útoku typu DNS spoofing, kdy bude klientovi podvržena falešná webová stránka (jednoduchý Phishing útok).

2.1 Realizace útoku - ARP Spoofing

Jak již bylo popsáno v teoretickém úvodu, viz kapitola 1.2, ARP protokol slouží k nalezení fyzické adresy na základě logické. Slabinou ARP protokolu je fakt, že byl navrhován v dobách, kdy bezpečnost nehrála při navrhování protokolu nehrála zásadní roli, a tudíž nemá žádné ochranné mechanismy.

Útok bude probíhat ve virtuální síti dle topologie v teoretickém úvodu 1.1. Topologie se skládá z **klienta**, **serveru** a **útočníka**. Tyto tři virtuální počítače společně mohou komunikovat skrze virtuální switch **VMware virtual switch**. Úkolem tedy bude modifikovat síť tak, aby všechna komunikace mezi klientem a serverem procházela skrze útočníka **(útok typu MITM)**.

Principem útoku bude odeslat klientovi falešný paket **ARP response**, ve kterém bude oznámeno, že server s IP adresou **192.168.12** má MAC adresu útočníka [**BB:BB:BB**]. Tím bude docíleno, že klient bude odesílat všechnu komunikaci směřující na server útočníkovi. Útočník tato data zachytí a následně přepošle na server.

Obdobně bude realizován i druhý směr toku dat. Serveru bude informován falešnou odpovědí **ARP response** o skutečnosti, že klient s IP adresou **192.168.1.10** má MAC adresu útočníka [**BB:BB:BB**].

V této úloze je třeba spustit následující virtuální počítače, které se opět nachází na ploše ve složce **Bezpečnost** s názvem **Útočník - Kali Linux 1.1.0**, **Server - Ubuntu Server 14.04.2 LTS** a **Klient - Xubuntu 14.04**

Po spuštění útočníka je nutné se přihlásit.

Útočník - Přihlašovací údaje: **Username:** `root`, **Password:** `kali`

Nyní se spustí příkazový řádek, pomocí kterého bude celý ARP spoofing realizován. Realizování ARP spoofingu je možné i snadnější cestou, například pomocí programu **Ettercap**, kde je možné využít grafické rozhraní. Pro větší názornost bude probíhat nastavení pouze pomocí terminálu.

Útočník - Je třeba spustit terminál, který lze nalézt ve vrchním panelu pod touto ikonou:  . Případně je možné nalézt terminál v nabídce **Applications/Accessories/Terminal**.

Jelikož bude nutné spustit několik oken, je zajímavým řešením využití nástroje **Screen**, který slouží jako správce sezení v terminálu a je schopný rozdělit terminálové okno na více různých sezení. Je však možné spustit terminál několikrát a přepínat mezi jednotlivými terminály.

Útočník (dobrovolné) - Pro spuštění screenu stačí v terminálovém okně zadat příkaz: `screen`. Zobrazí se základní informace o programu, dále se pokračuje stisknutím **Enter**. Nyní je **screen** aktivní a je možné rozdělit terminál pomocí níže uvedených zkratek.

Horizontální rozdělení
: V aktivním okně terminálu je nutné stisknout klávesovou zkratku `CTRL+A` a poté stisknout klávesovou zkratku `SHIFT+S`. Nyní by měl být terminál rozdělen horizontálně.

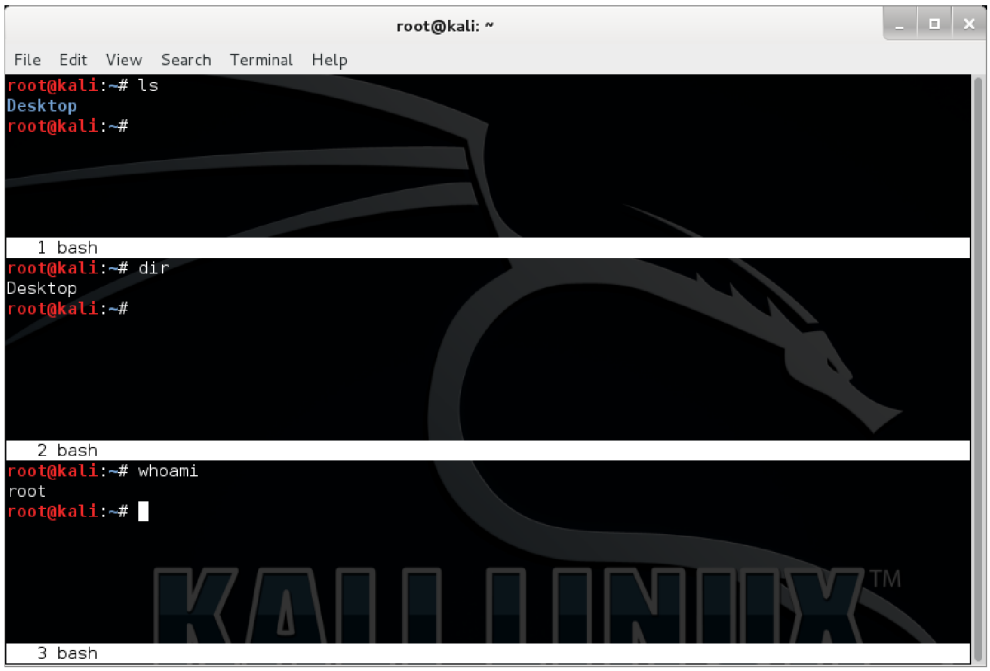
Vertikální rozdělení
: V aktivním okně terminálu je nutné stisknout klávesovou zkratku `CTRL+A` a poté stisknout klávesu „roura“ `|`. Nyní by měl být terminál rozdělen vertikálně.

Přepínání mezi jednotlivými sezeními
: V aktivním okně terminálu je nutné stisknout klávesovou zkratku `CTRL+A` a poté stisknout klávesu `TAB`.

Výběr sezení
: V aktivním okně terminálu je nutné stisknout klávesovou zkratku `CTRL+A` a poté stisknout klávesu `C`.

V případě zájmu je níže popsáno vhodné rozdělení sezení pro tuto úlohu. Jakkoli se může zdát ovládání programu **screen** ze začátku složité, tak se jedná o zajímavý program ulehčující přehlednost a práci s více terminály. Nastavení dle obrázku 2.1:

1. Horizontální rozdělení
2. Horizontální rozdělení
3. Přepnutí mezi jednotlivými sezeními a jejich následná aktivace



Obr. 2.1: Rozdělení terminálu pomocí programu screen

Protože tento typ útoku využívá přeposílání paketů skrze útočníka, je v první řadě nutné přepnout síťovou kartu tak, aby podporovala přeposílání. Toho lze docílit pomocí změny hodnoty v souboru **ip_forward** z `0` na `1`:

Útočník - Příkaz: `echo '1' > /proc/sys/net/ipv4/ip_forward`

Nyní se spustí příkaz **arpspoof**, pomocí kterého budou generovány falešné ARP odpovědi serveru a klientovi. Struktura příkazu je následující:

```
arpspoof [-i interface] [-t target] host
```

kde `-i` je síťové rozhraní, které je využito pro spoofing (v případě ethernetu např. eth0). Parametr `-t` specifikuje IP adresu cíle, kterému chce útočník zaslat falešnou ARP odpověď (v případě nevyplnění tohoto parametru bude odeslán všesměrově). Parametr `host` určuje adresu hostitele, kterému chce útočník odchytávat příchozí síťový provoz.

V případě této úlohy je na základě uvedené topologie nutné zadat následující hodnoty pro zachycení dat **od klienta směrem k serveru**:

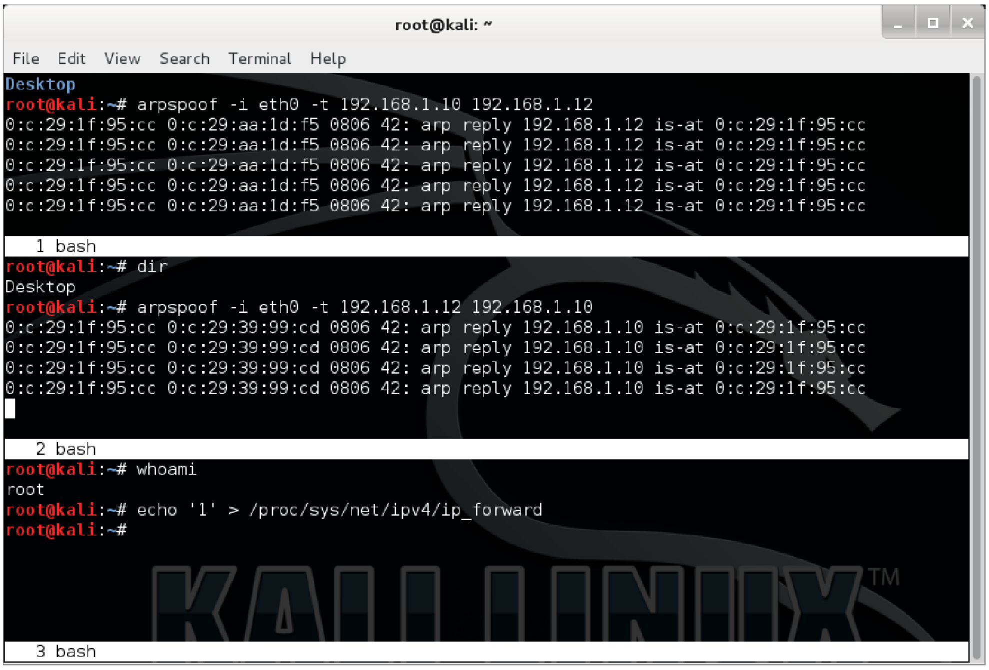
Útočník - Příkaz: `arpspoof -i eth0 -t 192.168.1.10 192.168.1.12`

Příkaz arpspoof začne opakovaně zasílat na cílovou IP adresu ARP request. **Je nutné nechat tento příkaz spuštěný** a následující příkazy spouštět buď v novém terminálovém okně, případně v jiném sezení programu **screen**.

Dále je třeba nastavit také druhý směr **od serveru ke klientovi**:

Útočník - Příkaz: `arpspoof -i eth0 -t 192.168.1.12 192.168.1.10`

Tímto je útok ARP spoofing kompletní. Skrze útočníka nyní prochází komunikace klient/server. Oba **arpspoof** příkazy pravidelně odesílají **ARP requests** na oba cíle. V případě využití programu screen bude vypadat okno terminálu následovně jako na obr. 2.2:



Obr. 2.2: Spuštěný arp spoofing

Nyní je třeba spustit zachytávání komunikace pomocí programu **Wireshark** obdobně jako v první části laboratorní úlohy. **ARP spoofing je stále zapnutý!**

Útočník - Spuštění programu **Wireshark**, který se nachází v hlavní nabídce **Applications/Internet/Wireshark**. Zapnutí zachytávání na rozhraní **eth0**

Dále je třeba vygenerovat provoz mezi klientem a serverem. Na klientovi se opět otevře webová aplikace `www.test-domena.cz`

Klient - Spustit webový prohlížeč a otevřít `www.test-domena.cz`. Vložit **libovolné** přihlašovací údaje a otestovat přihlášení.

Nyní se opět přepne na virtuální počítač útočníka a prozkoumají se zachycená data. Jelikož útočník na rozhraní **eth0** odesílá velké množství **ARP odpovědí**, budou tyto pakety i v zachycených datech. Je tedy možné opět využít jednoduchého filtru:

Útočník - Zastavit zachytávání paketů v programu **Wireshark**. Odfiltrovat ARP odpovědi pomocí filtru: `not arp`.

V zachycené komunikaci se nacházejí stejné informace jako v případě první části laboratorních cvičení. Pomocí níže uvedeného filtru je možné opět zobrazit odeslané heslo pomocí metody **POST**.

Útočník - Modifikovat stávající filtr na: `http.request.method == POST`

Pokud byly všechny kroky správně splněny, je zachyceno heslo. Opětovným otevřením detailů u zachyceného paketu s metodou **POST** se lze propracovat k zaslanému uživatelskému jménu a heslu.

Jak je vidět z výše uvedeného postupu, odchycení hesla a osobních údajů je v případě využití HTTP a plain textu velmi snadné. V žádném případě tedy nelze doporučit odesílat citlivé údaje pomocí tohoto protokolu a lze doporučit využití zabezpečené verze **HTTPS**. Při využití HTTP lze případně doporučit využít metodu pro autentizaci **HTTPS Digest**, kde není heslo posíláno v plain textu ale je odesílán otisk hesla spojený s náhodnými daty za pomoci hašovací funkce.

2.2 Realizace útoku - DNS Spoofing

Z teoretického úvodu, viz 1.3, je zřejmé, že primárním účelem DNS systému je zajištění překladu doménových jmen na IP adresy a naopak. DNS resolver (Klient) zasílá dotazy na DNS server, který má nastaven v nastavení sítě. Poté co získá odpověď, uloží si jí obdobně jako u ARP protokolu do DNS Cache pro opětovné rychlé použití.

Doba, po kterou bude uložen záznam v DNS Cache, je stanovena v odpovědi DNS serveru. Po vypršení stanovené doby je záznam smazán a je opětovně proveden překlad.

Principem DNS spoofingu je zjednodušeně zfalšování DNS odpovědi na DNS dotaz. Vzhledem k tomu, že DNS využívá standardně UDP protokol, je snadnější falešnou odpověď generovat. Není třeba zjišťovat sekvenční čísla apod.

Na druhou stranu je pro vygenerování DNS odpovědi potřebné znát následující informace:

IP adresa DNS serveru
: Je nutné zjistit IP adresu DNS serveru, kterou používá oběť.

ID dotazu
: Při generování dotazu DNS resolver (klient) generuje 16 bitové číslo. Toto číslo se musí nacházet i v odpovědi od serveru, pokud je číslo odlišné, resolver paket ignoruje.

Dotazované doménové jméno
: Pro zacílení útoku je vhodné znát dotazované doménové jméno.

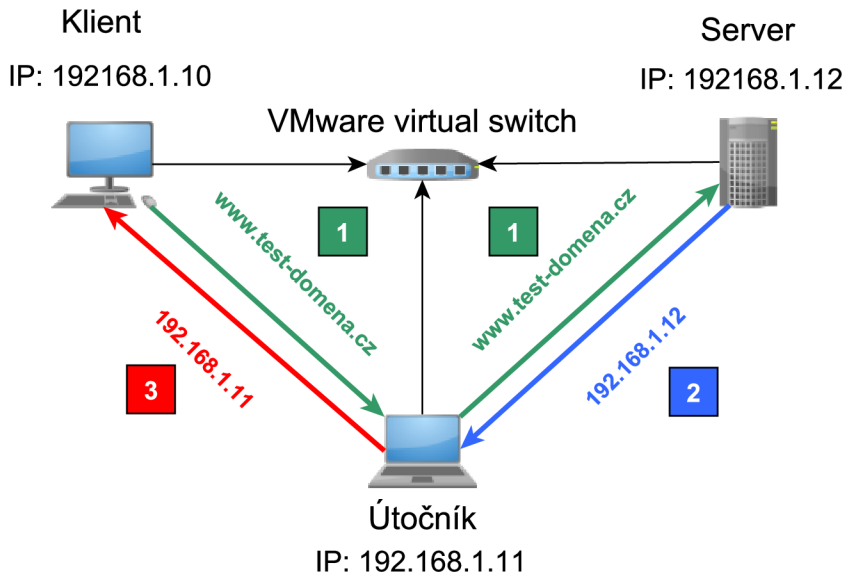
Port pro zaslání odpovědi na DNS resolver
: Falešnou odpověď je třeba zaslat na očekávaný port.

Ačkoli se může zdát, že je potřeba mnoho informací, tak většina DNS resolverů nepožaduje všechny výše uvedené informace. Stěžejní informací je **ID dotazu** a **číslo portu DNS resolveru**.

S výhodou lze využít probíhající **ARP spoofing** , nad kterým bude spuštěn DNS spoofing. Jelikož všechna komunikace probíhá skrze **útočníka**, nebude problém zachytit požadované informace. Průběh útoku bude následující:

1. Útočník bude zkoumat DNS dotazy od resolveru. V případě, že se bude jednat o dotaz na doménu, která má být podvržena (např. www.test-domena.cz), tak z paketu získá potřebné informace (vygenerované ID, Číslo portu) a propustí dotaz dále na server.
2. Odpověď od serveru v podobě přeložené IP adresy (např. 192.168.1.12) nebude propuštěna skrze útočníka.
3. Útočník vygeneruje falešnou DNS odpověď (např. IP adresa 192.168.1.11) na základě informací získaných v kroku 1 a zašle jí DNS resolveru (klientovi) na požadovaný port.

Celkový průběh útoku je znázorněn na následujícím obrázku 2.3:



Obr. 2.3: Průběh DNS spoofingu

Nyní se přistoupí k samotnému útoku. Je nutné zdůraznit, že **ARP spoofing z minulé části musí stále probíhat**. Pokud byl ukončen, je nutné znovu spustit generování ARP odpovědí v obou směrech! V této části bude využit příkaz **dnsspoof**, který má následující strukturu:

```
dnsspoof [-i interface] [-f hostsfile] host
```

kde `-i` určuje rozhraní, pomocí kterého má být použit (v tomto případě **eth0**). Parametr `-f` určuje cestu k souboru s doménovými jmény a IP adresy, které má v úmyslu útočník podvrhnout. Poslední parametr `host` určuje IP adresu oběti, které je cílem zaslat falešnou DNS odpověď (pokud bude pole host ponecháno prázdné, bude dnsspoof odpovídat na DNS dotazy od všech stanic).

Před spuštěním dnsspoof příkazu je tedy nutné vytvořit soubor s doménovými jmény a IP adresy. To je možné pomocí následujícího příkazu:

Útočník - Vytvoření souboru příkazem: `touch /home/falesneDNS`

Nyní je možné soubor editovat využitím textového editoru **vim** za pomoci příkazové řádky.

Útočník - Nyní proběhne editace souboru **falesneDNS** příkazem: `vim /home/falesneDNS`. V programu **vim** je možné editovat až po stisknutí klávesy **insert**

Nyní se v prostředí vim napíší požadovaná doménová jména a k nim odpovídající IP adresy. Cílem bude přesměrovat klienta při zadání doménového jména `www.test-domena.cz` na IP adresu útočníka `192.168.1.11`. V prostředí **vim** lze zahájit editaci souboru stisknutím tlačítka **insert**. Obsah cílového souboru bude vypadat následovně:

192.168.1.11 www.test-domena.cz

V souboru však může být hned několik záznamů. Na každém řádku může být jiný záznam. Doménové adresy v souboru lze zobecňovat, pokud by bylo třeba účelem přesměrovat provoz ze všech koncovek **www.test-domena** jako např. **cz**, **net**, **com**, **org** a další, je možné využít následující zobecnění pomocí hvězdičky:

```
192.168.1.11 www.test-domena.*
```

Útočník: Nyní se soubor uloží a ukončí editor stisknutím klávesy **Esc** a následným zadáním `:wq`. [1] (parametr `w` = write, tedy zapsání změn a parametr `q` = quit, ukončení editoru).

V této fázi je možné spustit samotný příkaz dnsspoof s následujícími parametry:

Útočník - Příkaz:
`dnsspoof -i eth0 -f /home/falesneDNS host 192.168.1.10`

Tímto je útok úspěšně spuštěný. Klient by měl být při zadání doménového jména **www.test-domena.cz** přesměrován na IP adresu **192.168.1.11**.

Ověřit funkčnost útoku lze na straně klienta využitím příkazu **ping**

Klient - Příkaz: `ping www.test-domena.cz`

Odpověď na příkaz ping by měla přicházet od IP adresy 192.168.1.11. Čímž lze považovat DNS spoofing za úspěšný.

K čemu je takovéto přesměrování užitečné, je možné ukázat na následujícím příkladu. Pokud klient zadá www.test-domena.cz, je přesměrován na útočníkův počítač. Zde se nabízí relativně snadná možnost spuštění kopie webového serveru na počítači útočníka a vytvoření jednoduchého **phishingu** ve formě falešných stránek.

Na virtuálním počítači útočníka se na ploše nachází modifikovaná kopie webové aplikace z předchozích úloh.

V prvním kroku se spustí webový server **Apache2** na počítači útočníka pomocí následujícího příkazu:

Útočník - Příkaz: `/etc/init.d/apache2 start`

Po úspěšném spuštění Apache2 je možné jeho běh ověřit otevřením webové stránky www.test-domena.cz ve webovém prohlížeči **na straně klienta**. Zobrazit by se měla testovací stránka Apache serveru „It works!“ (Pokud není možné zobrazit stránku s touto hláškou, je třeba restartovat webový prohlížeč).

Dále se přistoupí k přepsání této defaultní stránky výše zmíněnou webovou aplikací. Tato aplikace je připravena na ploše útočníka ve složce **dnsSpoof**. Nyní se překopírují soubory webové aplikace do složky pro Apache server:

Útočník:

1. V prvním kroku se smaže původní testovací soubor pomocí příkazu: `rm /var/www/index.html`
2. V dalším kroku dojde k překopírování složky se vzhledem pomocí příkazu: `cp -avr /root/Desktop/dnsSpoof/css/ /var/www`
3. Dále bude zkopírován soubor pro uložení hesel data.txt. Příkaz: `cp /root/Desktop/dnsSpoof/data.txt /var/www`
4. Dalším souborem ke zkopírování je samotný soubor index.php. `Příkaz:` `cp /root/Desktop/dnsSpoof/index.php /var/www`

Dále je nutné opravit práva zkopírovaných souborů pomocí následujících příkazů:

Útočník:

1. Oprava práv souborů v adresáři www na 775. Tímto bude umožněno zobrazit webové na straně klienta.
   Příkaz: `chmod -R 775 /var/www`

2. Přidání možnosti zápisu do souboru data.txt. Tímto je přidán plný přístup pro zápis, čtení a spouštění souboru, do kterého budou zachytávána uživatelská hesla.
   Příkaz: `chmod 777 /var/www/data.txt`

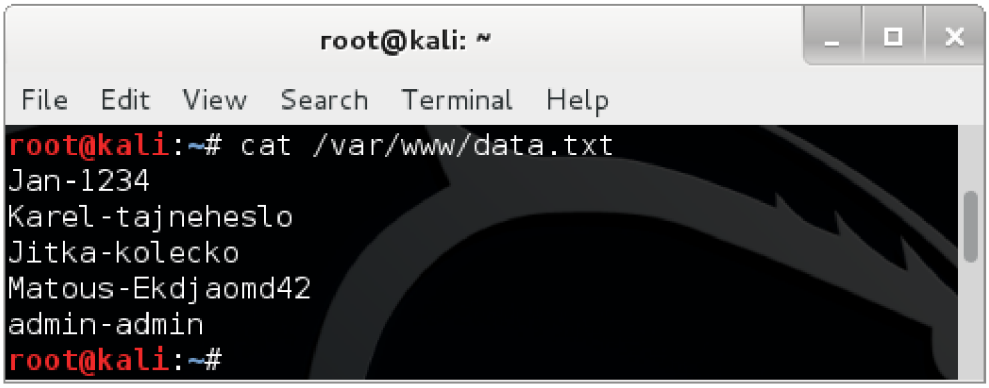
Tímto je nastavení kompletní, pokud si klient otevře webovou stránku www.test-domena.cz, je přesměrován na IP adresu útočníka (192.168.1.11), kde se nachází již funkční kopie webové aplikace, která má za účel sběr uživatelských jmen a hesel. Tyto informace ukládá do souboru **data.txt** umístěném v **/var/wwww**. Nyní je možné celý mechanismus otestovat

Klient - otevřít webový prohlížeč s adresou `www.test-domena.cz` a zadat několik libovolných **uživatelských jmen** a **hesel**

Nyní je možné zachycená uživatelská jména a hesla zobrazit na počítači útočníka pomocí příkazu `cat`. Tento příkaz slouží k vypsání obsahu souboru:

Útočník - Pro vypsání zachycených uživatelských jmen a hesel stačí zadat příkaz: `cat /var/www/data.txt`

Uživatelská jména a hesla jsou přehledně vypsána v příkazové řádce obdobně jako na obrázku 2.4.



Obr. 2.4: Vypsaná uživatelská jména a hesla

V tomto případě je účelem phishingu získávat pomocí jednoduché kopie webové aplikace pouze uživatelská jména a hesla. Bylo by však možné vytvořit i mnohem komplexnější kopie webových aplikací, což by mohlo mít rozsáhlejší následky.

2.3 Úkol

Cílem samostatného úkolu je modifikovat útok tak, aby byl klient přesměrován na falešnou webovou aplikaci i při zadání doménového jména **www.test-domena.com** a **www.vutbr.cz**(v současné době je přesměrována doménová adresa pouze **www.test-domena.cz**).

3 ZÁVĚR

V rámci druhé části laboratorní úlohy byl proveden útok typu MITM pomocí ARP spoofingu. Tímto bylo potvrzeno, že pokud útočník vnikne do komunikačního kanálu, kde je přenášen protokol HTTP, vznikne nebezpečí úniku citlivých informací. Dalším útokem, který byl postavený na ARP spoofingu, byla jednoduchá forma phishingu realizovaná DNS spoofingem. Za pomoci falešné webové aplikace bylo opět získáno heslo a uloženo v přehledné podobě do txt souboru.

U phishingu je však mnohem větší potenciál. V případě kvalitní kopie známého webu a kombinace sociálního inženýrství se jedná o velmi účinnou zbraň, pomocí které by bylo možné z oběti získat cílené informace.

Obranou proti ARP spoofingu je využití softwaru pro její detekci, případně statické přidělení IP adres k MAC adresám u ARP protokolu. Ochrana proti DNS spoofingu je obdobná jako u ARP spoofingu, dá se řešit detekčním softwarem, případně statickým zadáním překladů do souboru hosts v operačním systému.

3.1 Kontrolní otázky

1. K čemu slouží ARP protokol?
2. Jaký účel má DNS cache?
3. Jakým způsobem by bylo možné komunikovat prostřednictvím internetu, pokud by nevznikl DNS systém?

LITERATURA

[1] DIMITRIOS, Tournas. DNS spoofing with "dnsspoof" on Linux. *Tournas Dimitrios Learning the web and more* [online]. 2011 [cit. 2015-05-18]. Dostupné z URL: <https://tournasdimitrios1.wordpress.com/2011/03/03/dns-spoofing-with-dnsspoof-on-linux/>.

[2] POSPÍCHAL, Petr. *Útoky v počítačových sítích* [online]. 2008 [cit. 2015-05-18]. Dostupné také z URL: <http://petr.pospichal.biz/PDS/utoky-v-pc-sitich.pdf>.

[3] Dnsspoof(8) - Linux man page. *Die.net* [online]. [cit. 2015-05-18]. Dostupné z URL: <http://linux.die.net/man/8/dnsspoof>.

[4] HALLER, Martin. Seriál Odposloucháváme data na přepínaném Ethernetu. *Lupa.cz* [online]. 2006 [cit. 2015-05-18]. Dostupné z URL: <http://www.lupa.cz/serialy/odposlouchavame-data-na-prepinanem-ethernetu/#ic=serial-box&icc=title>.

[5] ZICHOVA, Hana. ARP POISONING. *Západočeská univerzita v Plzni* [online]. 2006 [cit. 2015-05-18]. Dostupné z URL: <http://home.zcu.cz/~zichovah/>.

D DOKUMENTACE PRO VYUČUJÍCÍHO - LABORATORNÍ ÚLOHA - ČÁST 2.

D.1 Základní informace

V této laboratorní úloze je realizován útok ARP spoofing pomocí počítače - **útočník**. V tomto kroku by si studenti měli uvědomit, že se nachází uprostřed komunikace klient/útočník. Dále je při spuštěném ARP spoofingu realizován útok DNS spoofing. Studenti by si měli uvědomit, že útočník kontroluje komunikaci klienta a odesílá klientovi falešné DNS odpovědi. V poslední části je spuštěn HTTP server apache, na kterém si studenti vyzkouší zprovoznění phishingové stránky pomocí připravené webové aplikace. Následně studenti zaznamenají uživ. jména a hesla klienta.

D.2 Řešení úkolu

Úkolem je modifikovat vytvořený soubor **falesneDNS**, který se nachází ve složce **home** na počítači útočníka, a modifikovat obsah souboru na následující:

```
192.168.1.11 www.test-domena.*
192.168.1.11 www.vutbr.cz
```

Poté se znovu spustí program **dnsspoof**

```
dnsspoof -i eth0 -f /home/falesneDNS host 192.168.1.10
```

Nyní je možné otestovat funkčnost pomocí webového prohlížeče.

D.3 Odpovědi na otázky

1. ARP protokol slouží k nalezení fyzické (MAC) adresy na základě IP adresy.
2. DNS Cache slouží pro uložení získaných DNS odpovědí od serveru. Účelem je urychlení následné komunikace, jelikož stanice nemusí opětovně provádět dotaz na DNS server pro stejný záznam.
3. Pokud by nevznikl systém DNS, tak by bylo nutné používat místo doménových jmen IP adresy, což by bylo pro člověka obtížné na zapamatování. Dalším řešením by bylo uložení rozsáhlého souboru **hosts** na každém počítači, který by obsahoval překlady.

E TEXT LABORATORNÍ ÚLOHY - ČÁST 3.

LABORATORNÍ ÚLOHA POČÍTAČOVÝCH ÚTOKŮ ČÁST 3.

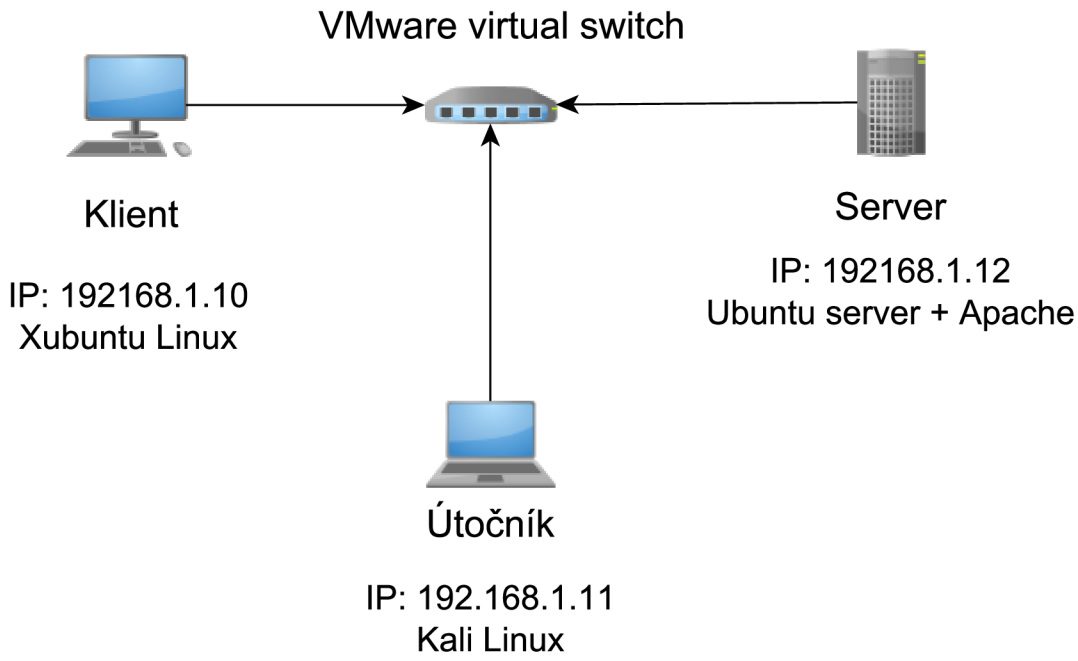
(Realizace síťového útoku s využitím Kali linux a útoku na webovou aplikaci)

Brno 2015

1 TEORETICKÝ ÚVOD

V rámci teoretické části třetí laboratorní úlohy proběhne seznámení s HTTPS protokolem. Dále budou popsány útoky na webové stránky Cross-site scripting (XSS) a SQL injection. V praktické části bude realizován útok SSLstrip, na kterém bude prezentováno narušení zabezpečené komunikace. V poslední části úlohy proběhne praktické otestování útoků XSS a SQL injection v připravené webové aplikaci. Úloha využívá níže uvedenou topologii 1.1.

1.1 Topologie sítě laboratorní úlohy



Obr. 1.1: Topologie sítě laboratorní úlohy

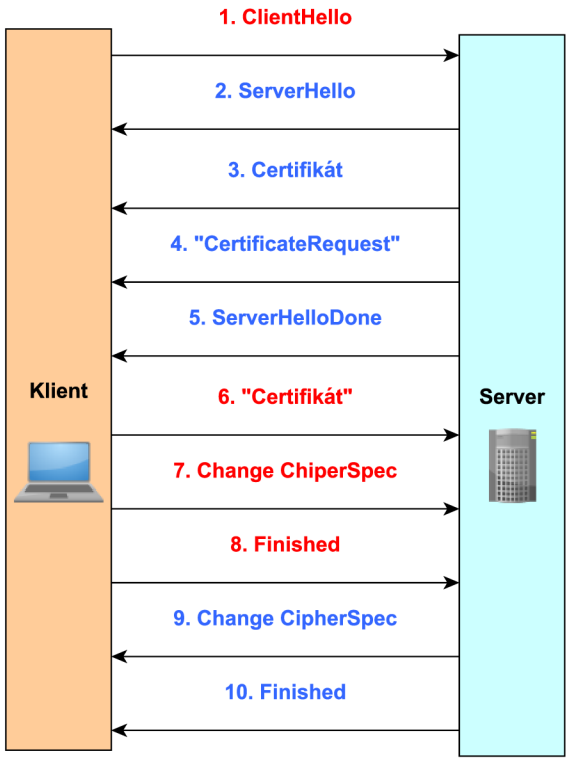
1.2 HTTPS protokol

V první části laboratorního cvičení (část 1.) byl popsán protokol HTTP, pomocí něhož je realizována komunikace mezi webovým prohlížečem (dále pouze klient) a serverem. Nedostatkem HTTP je „otevřenost“ protokolu. Pokud se kdokoli dostane mezi server a webový prohlížeč, je schopný číst komunikaci včetně hesel (s vyjímkou HTTP Digest) a dalších citlivých údajů, jako jsou například formuláře.

Aby se zabránilo snadnému odposlechu, vznikl protokol HTTPS (HyperText Transfer Protocol Secured). Protokol HTTPS využívá protokolu HTTP s tím, že jsou data před odesláním šifrována pomocí SSL nebo TLS protokolu.

1.2.1 Sestavení bezpečného spojení (SSL Handshake)

Níže je popsán základní princip při navazování spojení s využitím HTTPS protokolu za pomoci SSL Handshake Protokolu. Sestavení spojení funguje na principu asymetrické šifry (nejčastěji RSA), viz obr. 1.2.



Obr. 1.2: Sestavení HTTPS spojení (SSL Handshake)

1. Klient odesílá serveru zprávu **ClientHello**, která oznamuje svojí verzi šifrovacího protokolu, dále náhodné číslo a seznam podporovaných šifrovacích sad a kompresních metod.
2. Server odpovídá zprávou **ServerHello**, která obsahuje zvolenou verzi šifrovacího protokolu, šifrovací a kompresní metodu, na základě informací od klienta.
3. Server zasílá svůj **certifikát**. Klient si tento certifikát ověří u důvěryhodné certifikační autority.
4. Server může volitelně zažádat klienta o zaslání certifikátu pro vzájemné ověření.

5. Server zasílá zprávu **ServerHelloDone**, ta signalizuje ukončení dohody na použitých mechanismech.
6. V případě že byl klient vyzván v bodě 4 k odeslání certifikátu, tak v tomto okamžiku zašle svůj certifikát.
7. Klient odešle zprávu **ChangeCipherSpec**, zde klient oznamuje že další zprávy bude odesílat zašifrované.
8. Klient odešle zprávu **Finished**, která je zašifrována veřejným klíčem serveru. Server dešifruje příchozí zprávu. V případě neúspěšného dešifrování se spojení ukončí.
9. Server také odešle zprávu **ChangeCipherSpec** stejně jako klient.
10. Server odesílá zašifrovanou zprávu **Finished**, klient analogicky jako server zprávu dešifruje a ověří.

Mezi hlavní přínosy HTTPS patří zajištění **důvěrnosti** - pokud útočník šifrovaná data odposlechne, tak není schopen v současné době zachycená data dešifrovat. Dále zajištění **autentičnosti** - přenášená data obsahují kontrolu integrity dat za pomoci MAC (Message Authentication Code), je také prováděno ověření zaslaného certifikátu serverem pomocí důvěryhodné certifikační autority. Ověřením certifikátu certifikační autoritou se eliminuje možnost provést útok MITM (Man in the middle).

1.3 Cross-site scripting (XSS) útok

Jedná se o útok na webovou aplikaci. Zkratka XSS označuje Cross-Site Scripting, to znamená skriptování napříč sítěmi. Tento typ zranitelnosti je známý již mnoho let, přesto se lze setkat s touto zranitelností ve více než v 80 % webových aplikací. Zranitelnost má několik podob, které jsou nejčastěji děleny na trvalé (persistentní) XSS, dočasné (Non-persistentní) XSS a Lokální (DOM-based) XSS.

Původ této zranitelnosti začal s využíváním JavaScriptu ve webových stránkách za účelem oživení obsahu. V dřívější době bylo možné na webových stránkách zobrazovat pouze statické informace. Zobrazit aktuální datum, čas nebo přistupovat k objektům na stránce a měnit dynamicky jejich obsah bylo v dřívější době nemožné. S implementací v podobě JavaScriptu přišla společnost Netscape. V současné době existují i další skriptovací jazyky, ale vývojáři využívají převážně JavaScript.

Způsoby jak vložit do stánky skripty jsou tři. První možností je vložení skriptu mezi tagy `<SCRIPT>` a `</SCRIPT>`. Tyto znaky se dají vložit do hlavičky nebo těla HTML dokumentu. Poté co prohlížeč narazí na tag `<SCRIPT>`, tak vykoná obsahující kód.

Ukázka takto vloženého skriptu:

```
<script type="text/javascript">
  <!--
    document.write("Nazdar světe!");
  //-->
</script>
```

Skript vypíše na webovou stránku text „Nazdar světe“. Uvnitř skriptu se nachází také HTML značky pro komentáře `<!-, //->` v případě, že by prohlížeč nepodporoval JavaScript. V tomto případě by se komentovaná část prováděla jako HTML a skript by se na webové stránce nezobrazil. V dnešní době existuje jen málo webových prohlížečů, které neumí pracovat s JavaScriptem.

Druhou možností je načtení JavaScriptového kódu z externího souboru. Tento způsob má výhodu, jelikož v případě rozsáhlého JavaScriptu se nestane kód stránky nepřehledným. Další výhodou je snadná možnost použití JavaScriptového kódu na více webových stránkách a v případě modifikace skriptu stačí editovat pouze jeden soubor ***.js** místo webových stránek. Ukázka vložení kódu:

```
<script src="myscript.js"></script>
```

Poslední variantou je vložení pomocí In-line skriptu, který se vkládá jako atribut události. Tento skript se vykoná jako reakce na nějakou událost. Touto událostí může být kliknutí myší na odkaz, přejetí kurzorem myši apod.

```
<a href="http://www.vutbr.cz" onmouseover="alert('Ahoj světe!')">
vutbr.cz</a>
```

Výše uvedený skript zobrazí vyskakovací okno s textem „Ahoj světe!“ v případě najetí kurzorem na odkaz „vutbr.cz“.

Lokální (DOM-based) XSS

Při práci se skriptovacím jazykem na straně klienta se využívá modelu DOM (Document Object Model). Tento model definuje jednotlivé objekty na webové stránce, jako jsou otevřená okna prohlížeče, odstavce textu, vložené rámy, obrázky, formuláře, tlačítka a další prvky vyskytující se v dokumentu. Model DOM má přesně stanovenou hierarchii a při přístupu k objektu je třeba projít celou touto hierarchií.

U Lokálního XSS je využito výše uvedeného modelu na straně klienta. Skript, který útočník vloží do webové stránky, se neposílá na server a provádí se u klienta. Z tohoto důvodu se nazývá Lokální XSS.

Například stránka, která získává informace z parametru URI a dále tyto data zpracovává, se v případě neošetření stává zranitelnou. Na níže uvedeném příkladu je ukázka skriptu webové stránky, který je náchylný na Lokální XSS útok:

```
<script>
 var pos=document.URL.indexOf("color=")+6;
 document.write("Vaše barva je:
"+document.URL.substring(pos,document.URL.length));
</script>
```

Skript vyčte hodnotu proměnné **color**, která byla zadána do URI a zobrazí ji na stránce. V případě, že však útočník zadá do proměnné kód, jako je např. `<script>alert('Ahoj Světe!');</script>`, výsledná stránka bude obsahovat níže uvedený kód, který se po načtení ihned provede.

```
Vaše barva je: <b><script>alert("XSS");</script></b>
```

Ne-persistentní XSS

S touto zranitelností se lze setkat na webových stránkách, které zpětně zobrazují vložené parametry ze strany uživatele jako odezvu. Jsou to například vyhledávací pole, chybové hlášení (např. 404 stránka nenalezena). V případě vyhledávání uživatel vloží hledaný výraz „např. XYZ“ a webová stránka provede vyhledávání v databázi apod. a reaguje zobrazením výsledků. Například „Hledaný výraz XYZ nebyl nalezen“. Pokud tento vstup není správně ošetřen, může útočník místo hledaného výrazu vložit skript jako například `<script>alert('Ahoj světe!');</script>`. Část výsledné stránky poté může vypadat následovně:

```
Na váš hledaný výraz <b><script>alert("XSS");</script></b>
bylo nalezeno <b>10</b> výsledků.
```

Skript je jednorázově proveden, ale nezůstává na serveru – od toho název ne-persistentní.

Persistentní XSS

Tento typ útoku je možné využít u webových stránek, kde je možnost vložení trvalého příspěvku na webovou stránku. V praxi to mohou být návštěvní knihy, komentáře apod. Jestliže vstup od uživatele není dostatečně kontrolován, případně upraven logikou na straně serveru, může dojít k vložení kódu do stránky natrvalo „persistentně“. Vložený skript se poté spustí každému návštěvníkovi dané webové

stránky. V níže uvedeném příkladu je znázorněno jak by se takový příspěvek mohl zobrazovat pro ostatní uživatele.

```
\textsc{}<b>Neo</b>: <br>Mů skript ukrytý v příspěvku
<script>alert('Ahoj světe!');</script>
```

Na straně klienta se tedy spustí vložený skript a v tomto případě dojde pouze k otevření vyskakovacího okna s textem „Ahoj světe!“. V reálném prostředí jsou však možnosti využití tohoto útoku obrovské, od přesměrování na phishingovou stránku až po ukradení identity např. v podobě získání informací z cookies.

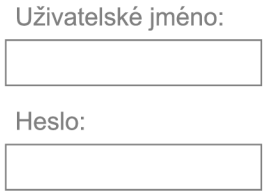
1.4 SQL útok

Jedná se o útok pracující na principu podvržení vstupních parametrů aplikační části webové stránky. Výsledkem správně zvolených parametrů bude spuštění SQL kódů nad databází, které není tvůrci webové stránky předpokládáno.

Jedná se o velmi nebezpečný útok, při kterém se útočník může dostat k datům uložených v databázi, jako jsou uživatelská jména nebo hesla (s největší pravděpodobností hash hesla). V horším případě je schopný pomocí SQL příkazů manipulovat s daty nebo i vymazat celou databázi. Ačkoli je tento útok známý, existuje stále velké množství webů, které nejsou proti tomuto útoku ošetřeny.

Neošetření vstupního formuláře

U webové stránky je zranitelné místo „neošetřený vstup“ v podobě přihlašovacího formuláře. Na níže uvedeném obrázku 1.3 je znázorněn typický přihlašovací formulář, který autentizuje uživatele na základě uživatelského jména a hesla.



Obr. 1.3: Přihlašovací formulář

Programová logika tohoto ověření by mohla být následující:

```
jmeno = getRequestString("UzivJmeno");
heslo = getRequestString("Heslo");
sql =
"SELECT * FROM Users WHERE Jmeno ='"+jmeno+"' AND Pass ='"+heslo+"'"
```

V případě, že uživatel do položky **UzivJmeno** a **Heslo** vloží například `"A" = "A"`, tak jeho výraz bude mít vždy logický výsledek `TRUE`. Tímto vznikne neošetřená situace a databáze na dotaz odpoví navrácením celého sloupce **Users** z databáze.

Neošetření vstupu z URI

Druhým příkladem je zneužití dotazů zasílaných na server URI. Níže uvedený kód má za úkol navrátit vyžádaný článek z databáze podle hodnoty `id` článku.

```
sql = "select * from clanky where id = '$_GET["id"]'"
```

Zavolání této funkce z prohlížeče může vypadat například takto:

```
http://www.domena.cz/clanek.php?id=1
```

Pokud útočník modifikuje dotaz v URI například na:

```
http://www.domena.cz/clanek.php?id=1 and "A"="A".
```

je docíleno stejného efektu jako v předchozím příkladu. Parametr je vždy pravdivý a webová stránka vypíše všechny hodnoty ze sloupce **clanky**.

Pokud útočník zjistí výše uvedené slabiny, může využít například příkazu `DROP`, který slouží ke smazání tabulky. Tímto krokem dojde k ochromení webové stránky. Pro zjištění zranitelnosti SQL injection lze využít jednoduchý nástroj **Havij** pro automatické skenování stránek.

2 PRAKTICKÁ ČÁST

2.1 Útok SSL strip a jeho realizace

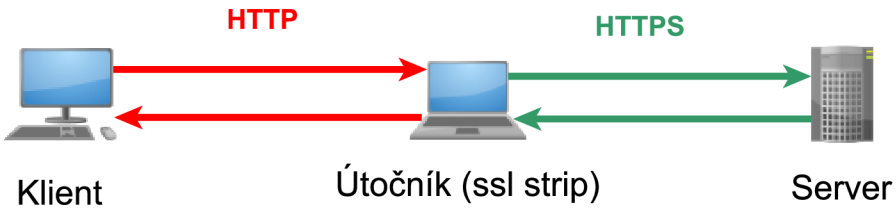
Protokoly HTTPS, SSL a TLS jsou v současné době považovány za bezpečné. Pokud nedojde k vyzrazení šifrovacích klíčů, případně není zvoleno velmi krátké heslo, tak není možné komunikaci pomocí HTTPS dešifrovat.

Existují však další způsoby útoků, které se nezaměřují na protokol HTTPS jako takový, ale na technologie s ním spojené, ty jsou mnohdy největší slabinou celého systému. Jedním z těchto útoků je také SSL strip.

V roce 2009 přišel Moxie Marlinspike s konceptem SSL stripingu. Ten vychází z myšlenky, že velké množství webových stránek má často úvodní stránku dostupnou s protokolem HTTP. Tato úvodní stránka poté obsahuje buď odkaz nebo přímo formulář pro přihlášení, který je přesměrován na HTTPS.

Pokud se útočník dostane mezi komunikaci klient/server (například pomocí ARP spoofingu), tak může zkoumat HTTP dotazy a s velkou přesností detekovat v datech odkazy směřující na HTTPS (například moment přihlášení). Na tomto principu pracuje SSL strip, který veškerá data s protokolem HTTP sleduje a všechny odkazy cíleně mění na HTTP. Zároveň si ukládá do mezi-paměti provedené změny.

Útočník tedy s klientem udržuje stále spojení HTTP (i přesto že mělo být za normálních okolností již HTTPS). Zároveň sestaví HTTPS spojení se serverem, viz obr. 2.1. Komunikace s klientem je plně transparentní, tudíž útočník může zachytit všechna hesla, odeslané formuláře apod.



Obr. 2.1: Schéma narušení HTTPS spojení (Útok SSL strip)

Tento typ útoku je v současnosti stále použitelný a v případě správného provedení je také těžko odhalitelný. Z takto napadnutelných webů lze jmenovat známé weby jako www.aukro.cz, www.centrum.cz, v případně i www.vutbr.cz.

Jak již schéma topologie sítě napovídá, je třeba využít k realizaci SSL stripingu tři virtuální počítače.

Nyní je třeba spustit tři virtuální počítače, které se opět nachází ve složce **Bezpečnost** na ploše. Je nutné spustit útočníka: **Útočník - Kali Linux 1.1.0**, server: **Server - Ubuntu Server 14.04.2 LTS** a klienta: **Klient - Xubuntu 14.04**.

Po spuštění útočníka platí níže uvedené přihlašovací údaje:

Útočník - Přihlašovací údaje: **Username:** `root`, **Password:** `kali`

Prvním krokem k realizaci SSL strip útoku je vytvoření již známého útoku MITM pomocí ARP spoofingu. Nejprve je třeba přepnout síťovou kartu tak, aby umožňovala přeposílání provozu pomocí níže uvedeného příkazu:

Útočník - Příkaz: `echo '1' > /proc/sys/net/ipv4/ip_forward`

Dále se spustí ARP spoofing, který bude generovat falešné ARP response. Příkaz pro odesílání falešných ARP odpovědí ke klientovi:

Útočník - Příkaz: `arpspoof -i eth0 -t 192.168.1.10 192.168.1.12`

Druhý příkaz pro odesíláné falešných ARP odpovědí k serveru:

Útočník - Příkaz: `arpspoof -i eth0 -t 192.168.1.12 192.168.1.10`

Nyní je cílem zajistit, aby útočník analyzoval protokol HTTP (Port 80) a v přenášeném HTML kódu měnil HTTPS odkazy na HTTP. K tomuto účelu bude využit nástroj **sslstrip**. Nejprve je však nutné přesměrovat příchozí HTML provoz na vstup nástroje **sslstrip**. SSL strip je možné nakonfigurovat tak, aby naslouchal na libovolném protokolu. Pro praktický příklad bude zvolen port **8080**. Nyní samotný příkaz pro přesměrování portů:

Útočník - Příkaz:

```
iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j
REDIRECT --to-port 8080
```

Nyní se provede spuštění nástroje **sslstrip**. SSLstip je relativně jednoduchý nástroj, který obsahuje několik základních voleb:

-w
: výstupem nástroje je log soubor, u kterého lze definovat cestu pomocí tohoto parametru.

-l
: specifikuje port na, kterém má sslstrip naslouchat.

-f
: sslstrip může zadáním tohoto parametru odesílat místo ikony (favicon) stránky obrázek zámku simulující indikaci HTTPS protokolu. Tím může docházet na straně klienta k přesvědčení, že jde o bezpečné připojení.

-k
: Po spuštění sslstrip je možné pomocí tohoto parametru ukončit všechna probíhající sezení (sessions).

Nyní je možné přistoupit ke spuštění sslstripingu. Adresa pro umístění log souboru bude `/root/Desktop/sslstrip`. Naslouchání aplikace bude nastaveno na **port 8080** (dle předešlé konfigurace iptables). Bude využit parametr **-k** pro ukončení všech probíhajících sessions. Konečný příkaz je zobrazen níže:

Útočník - Příkaz:

```
sslstrip -w /root/Desktop/sslstrip -l 8080 -k
```

Tímto je útok aktivní, nyní je potřeba pro zachycení dat generovat provoz na klientské stanici.

Klient - Nyní je třeba na klientském počítači otevřít webový prohlížeč s adresou `www.test-domena.cz/sslstrip.html`

Po načtení se zobrazí vytvořená testovací aplikace, která simuluje problém již zmíněných webových stránek. Otevřená webová stránka používá protokol HTTPS. Po kliknutí na odkaz pro přihlášení směřuje odkaz na zabezpečenou verzi https://www.vutbr.cz.

Jelikož je však sslstrip útok aktivní, nedojde při kliknutí na přihlášení k přesměrování na HTTPS. Pokud uživatel není pozorný, nemusí si všimnout, že je na nezabezpečené stránce a přihlásí se.

> **Klient** - Nyní je třeba kliknout na **odkaz pro přihlášení**, poté na přihlašovací stránce zvolit náhodné přihlašovací uživatelské jméno/heslo a **pokusit se o přihlášení**.

Po otestování přihlášení je třeba se opět přepnout na virtuální počítač útočníka. Jelikož klient vygeneroval provoz (přesněji se pokusil o přihlášení), bude zachycené uživatelské jméno a heslo uloženo v log souboru sslstripu. Nyní je možné prohlédnout log například pomocí příkazu:

> **Útočník** - Příkaz: `cat /root/Destop/sslstrip`

Výstup by měl být obdobný jako na níže uvedeném obrázku 2.2.

```
root@kali:~# cat /root/Desktop/sslstrip
2015-05-12 10:38:51,516 POST Data (www.test-domena.cz):
username=uzivatelTest&password=tajneHeslo&submit=+P%C5%99ihl%C3%A1sit+se+
root@kali:~#
```

Obr. 2.2: Odchycené uži. jméno a heslo pomocí SSLstrip

Tímto je SSLstrip útok úspěšně dokončený. Nyní je třeba probíhající útok ukončit, a to jak ARP spoofing, tak i SSLstrip. A restartovat virtuální počítač útočníka.

> **Útočník** - Nyní je třeba ukončit probíhajících útok pomocí zavření terminálového okna, případně klávesovou zkratkou **CTRL+C** a počítač **vypnout** pomocí příkazu: `poweroff`

2.2 Realizace útoku - Cross-site scripting (XSS)

V této části bude realizován útok Cross-site scripting. K útoku budou potřeba virtuální počítač: **Klient - Xubuntu 14.04** a **Server - Ubuntu 64bit**, útočník zůstává vypnutý.

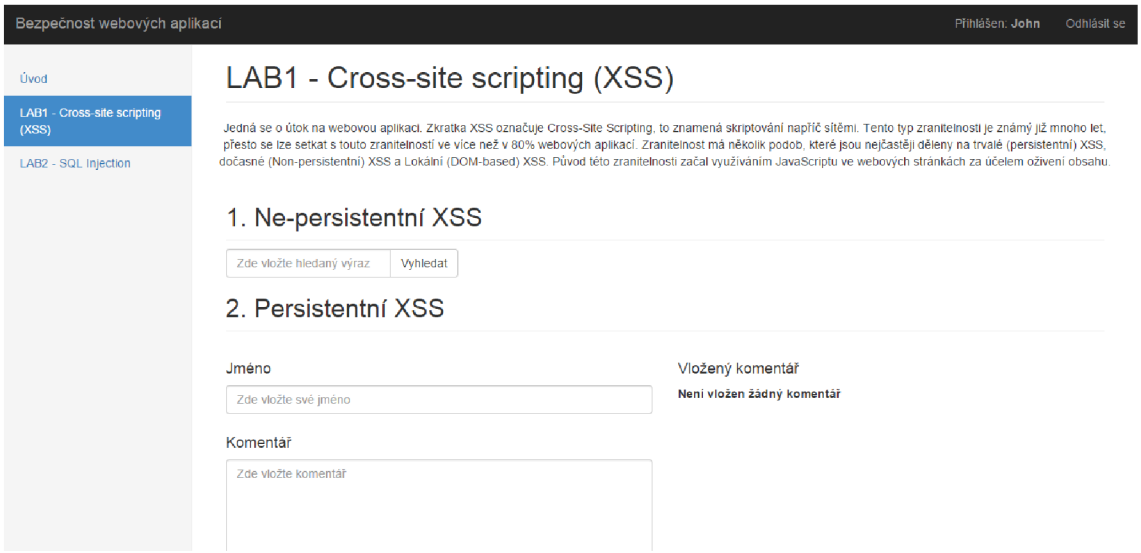
Útoky budou probíhat na testovací webové aplikaci, která je dostupná na již známé adrese `www.test-domena.cz`. Jedná se o aplikaci, která záměrně obsahuje neošetřené vstupy a je pomocí aplikace možné útok demonstrovat. Do aplikace je nutné se přihlásit podle zobrazených informací.

Klient - Nyní je třeba ve webovém prohlížeči otevřít webovou aplikaci, která se nachází na adrese `www.test-domena.cz` a přihlásit se s **uživatelským jménem** `John` a **heslem** `1234`.

Nyní se vybere první úloha LAB1, která je zaměřená na útoky XSS.

Klient - Po přihlášení vybrat z levého panelu **LAB1 - Cross-site scripting**

Po zadání přihlašovacích údajů a výběru úlohy je zobrazeno základní prostředí, ve kterém budou útoky simulovány viz obr 2.3.



Obr. 2.3: Spuštěná úloha

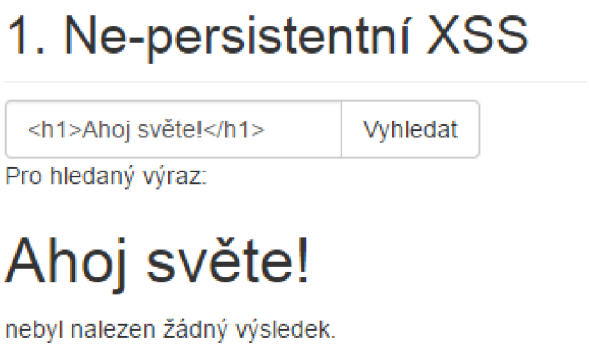
Nyní je možné přejít k útoku samotnému.

1. Ne-persistentní XSS

Jako první testovaný útok bude ne-persistentní XSS, který byl popsán v teoretickém úvodu. Ve webové aplikaci je připravený formulář pro vyhledávání. Pokud uživatel vloží do pole „zde vložte hledaný výraz“ text (např. `Ahoj světe!`) a potvrdí, tak dojde k odeslání textu na server. Po zpracování textu serverem dojde k navrácení výsledku v podobě „Pro hledaný výraz: **Ahoj světe!** nebyl nalezen žádný výsledek.“.

Klient - Nyní se otestuje vstup pro vyhledávání vložením **libovolného** textu. Následně dojde k otestování zabezpečení vstupu vložením HTML tagů okolo vyhledávaného textu. Pro otestování vstupu pomocí HTML tagů je možné vložit například výraz: `<h1>Ahoj světe!</h1>`.

Pokud byl vstup zadán správně, tak při vložení vstupu s HTML tagy, byly tyto tagy také zpracovány, výsledek by měl být obdobný jako na obr. 2.4. To znamená, že tento vstup není ošetřen na vložené znaky `<,>`. Výstupní text se stal nadpisem první úrovně H1.

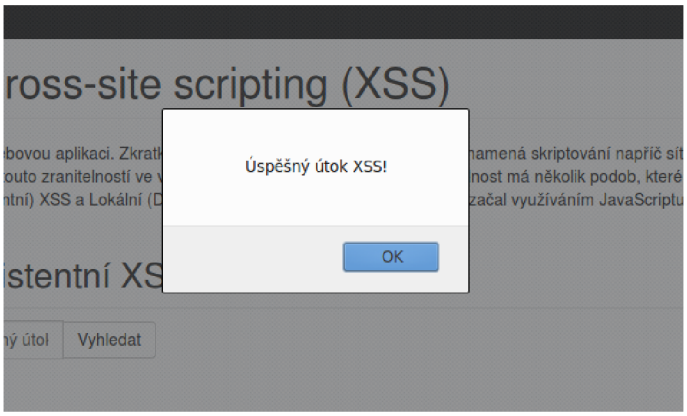


Obr. 2.4: Vložený text s tagy <h1>

V této části je také vhodné otestovat tagy `<script>,</script>`, které určují přítomnost skriptu na stránce. To bude provedeno vložením jednoduchého Javascript výrazu, který zobrazí vyskakovací okno s upozorněním. Nyní je třeba vložit do vyhledávacího pole následující výraz:

Klient - Do vyhledávacího pole je třeba vložit výraz: `<script>alert("Úspěšný útok XSS!");</script>`.

Pokud byl vložen výraz správně, dojde k zobrazení vyskakovacího okna s textem „Úspěšný útok XSS!“ stejně jako na níže uvedeném obrázku 2.5.



Obr. 2.5: Úspěšné spuštění Javascriptu

Dále je možné vyzkoušet Javascript kód pro zobrazení cookie dokumentu, které zobrazí session ID přihlášeného uživatele (John).

Klient - Otestování Javascript kódu pro zobrazení cookie: `<script>alert(document.cookie);</script>`.

Případně je možné otestovat Javascript kód pro přesměrování.

Klient - Otestování Javascript kódu pro přesměrování na úvodní stránku: `<script>window.location.replace("http://www.test-domena.cz");</script>`.

2. Persistentní XSS

Nyní je třeba se opět z úvodní stránky přepnout na LAB1. Dále bude otestován persistentní XSS, který je popsán v teoretickém úvodu, viz 1.3.

Pro testování persistentního XSS je využito ve webové aplikaci formuláře s neošetřeným vstupem. Jedná se o typický formulář pro vkládání komentářů. V momentě, kdy uživatel zadá **Jméno** a **Komentář**, dojde k odeslání těchto dat na server. Tyto informace se na serveru uloží, při každém načtení stránky jsou data nahrána. Nyní je třeba zadat náhodné údaje pro otestování funkčnosti formuláře:

Klient - Do formuláře se vloží libovolné **jméno** a **komentář**. Po stisknutí na tlačítko vložit komentář dojde ke vložení komentáře, který se následně zobrazí na pravé straně stránky.

Stisknutím klávesy **F5** je vhodné se přesvědčit, že komentář zůstává na stránce, dokud není přepsán vložením nového komentáře, případně vymazán tlačítkem **vymazat komentář**. Zde je důležité upozornit, že kliknutím na název úlohy v levém panelu způsobí načtení defaultního nastavení úlohy, a tím vymazání všech zadaných dat!

Nyní se obdobně jako u ne-persistentního útoku otestuje vstup pomocí níže uvedených vstupních dat. U níže uvedených skriptů je vhodné otestovat jejich persistenci na stránce pomocí **F5**.

Klient - Nyní je možné otestovat zabezpečení vstupu vložením libovolného jména a komentáře např: `<h1>Ahoj světe</h1>`. Text Ahoj světe je opět zobrazen nadpisem první úrovně H1.

Klient - Vložení komentáře:
`<script>alert("Úspěšný útok XSS!");</script>`.

Klient - Vložení komentáře: `<script>alert(document.cookie);</script>`.

Klient - Vložení komentáře:
`<script>window.location.replace("http://www.test-domena.cz");`
`</script>`.

3. Lokální (DOM) XSS

Posledním popsaným typem XSS útoku je lokální (DOM) XSS. Jedná se o lokální XSS útok. Jak již bylo popsáno v teoretickém úvodu, tento útok využívá DOM (Document Object Model) model.

Pro snadnější pochopení je ve třetí části XSS vytvořena jednoduchá funkce, která dle zadaného jména v URI vypisuje pozdrav. Pokud je v URI například `www.test-domena.cz/ulohy/lab1.php?jmeno=John`. Hodnota parametru **jmeno** je `John` a na stránce se vypíše **Vítej John**.

Zpracování URI probíhá na straně klienta pomocí skriptu obsaženém v zaslané HTML stránce, server tento parametr nezpracovává. Pokud nebyl tento vstup pomocí parametru z URI ošetřen, bylo možné pomocí něho opět spustit Javascript.

V dnešní době je však ochrana implementována v samotném prohlížeči, proto tento útok slouží pouze jako simulovaná ukázka a je nutné zadávat přesně níže uvedené vstupy.

Pro zobrazení vyskakovacího okna je nutné zadat do parametru následující skript:

Klient - Vložení URI parametru:
`jmeno=<script>alert("XSS");</script>`.

Pro zobrazení session ID v cookies prohlížeče:

Klient - Vložení URI parametru:
`jmeno=<script>alert(document.cookie);</script>`.

Tímto by měl být úspěšně proveden simulovaný lokální DOM XSS.

Ačkoli se výše testované skripty mohou zdát jako relativně neškodné, jedná se o velmi nebezpečnou slabinou. Lze si například představit umístění skriptu do komentáře na stránce `www.aaa.cz` , který by odkazoval na konkurenční stránku `www.bbb.cz`, případně na phishingovou stránku.

2.3 Realizace útoku - SQL injection

Pokud webová stránka pracuje s databází, tak většinou využívá určitý programovací jazyk, pomocí něhož s ní komunikuje, příkladem může být rozšířený jazyk PHP.

Př: Uživatel načte stránku pro přihlášení. Vyplní přihlašovací formulář, tyto informace jsou zaslané serveru, server na své straně spustí PHP skript, kde jsou vstupem uživatelem zaslané údaje. V PHP skriptu se nachází vytvoření dotazu na databázi se vstupními daty od uživatele. Pokud dotaz nalezne uživatele, je například přihlášen.

Problém zde nastává v PHP skriptu. Pokud skript není řádně ošetřen, tak je možné zadat takové vstupní informace, že je možné dotaz na databázi modifikovat a vytvořit tím například naprosto odlišný dotaz.

Nyní je třeba se přesunout na danou laboratorní úlohu **LAB2 -SQL Injection** ve webové aplikaci.

Klient - Otevřít **LAB2 -SQL Injection** ve webové aplikaci

1. Neošetřený vstupní formulář

V první části se jedná o jednoduchý vstupní formulář, pomocí kterého si mohou zaměstnanci fiktivní firmy zkontrolovat hodnotu své výplatní pásky pro daný měsíc. Autentizují se jednorázovým kódem, který je náhodně vygenerován.

Nyní se ověří funkčnost zadáním kódu, který vlastní fiktivní zaměstnanec Jiří Kotrba:

> **Klient** - Do položky **vložte zaměstnanecký kód** se vloží následující kód `A3hHXn6532` a potvrdí.

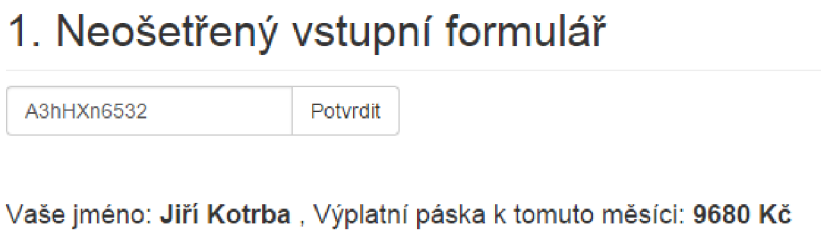
Nyní proběhne nalezení hodnoty výplatní pásky dle zadaného kódu pomocí níže uvedeného dotazu, kde proměnná `$id` obsahuje zadaný kód:

```
$query = "SELECT * FROM sqlinjection WHERE zamKod ='" . $id . "'";
```

po doplnění proměnné je výsledný dotaz následující:

```
$query = "SELECT * FROM sqlinjection WHERE zamKod ='A3hHXn6532'";
```

Provádí se tedy vyhledávání řádku v tabulce sqlinjection, kde je zamKod rovný hodnotě A3hHXn6532. V případě nalezení bude výsledek obdobný jako na níže uvedeném obrázku 2.6. V případě, že položka není nalezena, je zobrazena hláška: `Byl zadán špatný zaměstnanecký kód!`



Obr. 2.6: Nalezení zaměstnaneckého kódu

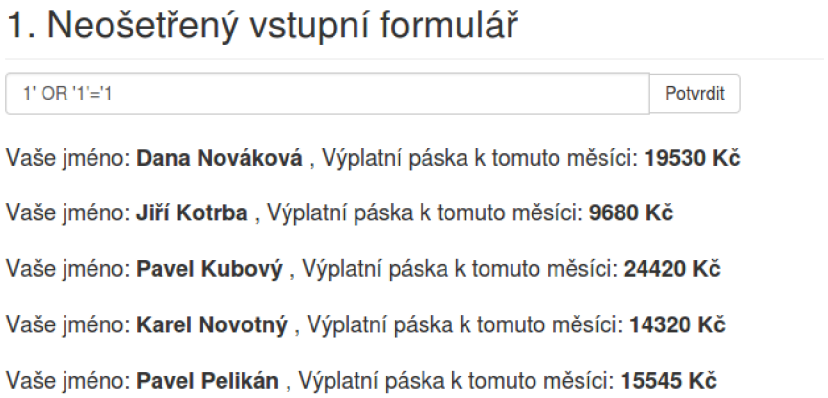
Jelikož vstup s vložením zaměstnaneckého kódu není nijak ošetřen, je možné do vstupu vepsat i kus kódu, a tím způsobit modifikaci sql dotazu.

> **Klient** - Do formuláře **vložte zaměstnanecký kód** se vloží následující kód: `1' OR '1'='1` Pozn: Není vhodné tuto proměnnou kopírovat, jelikož apostrofy nejsou správně zkopírovány. Znak „apostrof“ je v ASCII tabulce 39 znak v DEC.

Tímto výše uvedeným kódem je způsobena modifikace dotazu sql. Po doplnění proměnné do dotazu je výsledný dotaz následující:

```
$query = "SELECT * FROM sqlinjection WHERE zamKod ='1' OR '1'='1'";
```

Provádí se tedy vyhledávání řádku v tabulce sqlinjection, kde je zamKod rovný hodnotě 1 a nebo kde je 1=1. Ačkoli žádný zaměstnanecký kód se nerovná 1, tak výraz 1=1 je vždy pravda a každý řádek je tedy platný. Následkem tohoto příkazu dojde k vypsání celé tabulky viz obr 2.7.



Obr. 2.7: Úspěšný SQL injection

Tímto jsou např. kompromitovány všechny informace o výplatních páskách zaměstnanců.

2. Neošetřený přihlašovací formulář

Nyní bude proveden útok na druhou část - neošetřený přihlašovací formulář. Přihlašovací formulář slouží k přihlašování zaměstnanců pomocí uživatelského jména a hesla. Nyní je možné opět ověřit přihlašování.

> **Klient** - Do přihlašovacího formuláře se vloží **uživatelské jméno:** `kotrbajiri` a **heslo:** `kolotoc98`

Nyní opět proběhne nalezení řádku, které odpovídá zadaným přihlašovacím údajům, kde proměnná `$username` obsahuje uživatelské jméno a `$password` heslo.

```
$query2 = "SELECT * FROM sqlinjection WHERE
uzivJmeno ='" .$username. "'AND heslo='" .$password. "'";
```

po doplnění proměnných je výsledný dotaz následující:

```
$query2 = "SELECT * FROM sqlinjection WHERE
uzivJmeno ='kotrbajiri'AND heslo='kolotoc98'";
```

Úspěšné přihlášení bude indikováno zobrazením textu: `Uživatel kotrbajiri byl úspěšně přihlášen.` Nyní proběhne opět útok, pomocí kterého budou vypsána uživatelské jména.

Klient - Do přihlašovacího formuláře se nyní vloží libovolné **uživatelské jméno:** například `aaa` a do **hesla:** `1' OR '1'='1`. I zde platí doporučení nekopírovat heslo, jedná se opět o znak v ASCII tabulce 39 znak v DEC

Nyní bude opět modifikován sql dotaz na následující:

```
$query2 = "SELECT * FROM sqlinjection WHERE
uzivJmeno ='aaa'AND heslo='1' OR '1'='1'";
```

Tomuto dotazu opět odpovídají všechny řádky a je vypsaná celá tabulka viz obr. 2.8.

Uživatel **novakovadana** byl úspěšně přihlášen.

Uživatel **kotrbajiri** byl úspěšně přihlášen.

Uživatel **kubovypavel** byl úspěšně přihlášen.

Uživatel **novotnykarel** byl úspěšně přihlášen.

Uživatel **pavelkajan** byl úspěšně přihlášen.

Obr. 2.8: Úspěšný SQL injection - Přihlášení uživatele

Informace o zobrazených uživatelských jménech lze zneužít a přihlásit určitého uživatele pomocí níže uvedené modifikace současného útoku. Nyní proběhne konkrétní přihlášení libovolného uživatele bez znalosti hesla

Klient - Do přihlašovacího formuláře se nyní vloží libovolné **uživatelské jméno:** například `aaa` a do **hesla:** `1' OR '1'='1' AND uzivJmeno='novakovadana`. I zde platí doporučení nekopírovat heslo, jedná se opět o znak v ASCII tabulce 39 znak v DEC.

Pokud byly zadány hodnoty správně, je nyní přihlášen uživatel **novakovadana**. Je však možné přihlásit libovolného existujícího uživatele.

Je nutné zdůraznit, že hesla v databázi této aplikace jsou v nezabezpečené podobě z důvodu demonstrace útoku. V drtivé většině webových aplikací je dnes uložen pouze otisk hesla a vkládané heslo je opět podrobeno haš funkci a poté porovnáno s hodnotou uloženou v databázi.

2.4 Úkol

Na základě získaných informací je nyní úkolem vložit takový vstupní kód do formuláře v části **1. Neošetřený vstupní formulář**, aby byl vypsán plat pouze jednoho určitého pracovníka (např. Pavel Kubový) . Vycházet je možné z popsaného kódu pro první část a již demonstrovaných vstupů.

3 ZÁVĚR

Ve třetí části laboratorního cvičení byl realizován útok SSLstrip. Pokud je webová stránka, která obsahuje přihlašovací formulář případně odkaz pro přihlášení pomocí protokolu HTTPS dostupná protokolem HTTP, vzniká možnost využití útoku SSLstrip. Řešením je využít šifrovaného spojení na všech stránkách, které odkazují na přihlašovací stránky či obsahují přihlašovací formuláře.

V další části bylo otestováno několik metod Cross-site scripting (XSS) útoků. Pomocí XSS útoků lze spustit zákeřný Javascript kód, který může například útočníkovi odeslat indormace z cookies oběti, případně oběť přesměrovat na phishingovou stránku. Řešením je zde ošetření vstupních formulářů metodou zvanou escaping.

Posledním testovaným útokem byl SQL injection, pomocí kterého lze modifikovat dotazy nad databází. Řešením této zranitelnosti je opět v ošetření vstupů obdobně jako u XSS. Příkladem řešení SQL injection útoku může být využití hašovací funkce na vstupní formuláře (pokud data v databázi nemusí být reprezentována v čitelné podobě).

3.1 Kontrolní otázky

1. Je možné odposlechem HTTPS přenosu získat citlivé informace (užv. jména, hesla, odeslané formuláře, apod)?
2. Jaký je rozdíl mezi persistentním a ne-persistentním XSS útokem?
3. Jaký je princip SSLstrip útoku?

LITERATURA

[1] KÜMMEL, Roman. Pokročilé techniky XSS. *Soom.cz* [online]. 2008 [cit. 2015-05-18]. Dostupné z URL: <http://www.soom.cz/clanky/485--Pokrocile-techniky-XSS>.

[2] Testing for Cross site scripting. *Owasp.org* [online]. [cit. 2015-05-18]. Dostupné z URL: <https://www.owasp.org/index.php/Testing_for_Cross_site_scripting>.

[3] KULMAN, Igor. SQL Injection pre každého. *Zdrojak.cz* [online]. 2014 [cit. 2015-05-18]. Dostupné z URL: <http://www.zdrojak.cz/clanky/sql-injection-pre-kazdeho/>.

[4] ŠEVEČEK, Ondřej. Proč je alza.cz špatně udělaný web z pohledu HTTPS bezpečnosti. *Sevecek.com* [online]. 2014 [cit. 2015-05-18]. Dostupné z URL: <http://www.sevecek.com/Lists/Posts/Post.aspx?ID=453>.

[5] SSL protokol. *SSL-cerfifikaty.cz* [online]. [cit. 2015-05-18]. Dostupné z URL: <https://www.ssl-certifikaty.cz/o-certifikatech/ssl-protokol/>.

[6] SSL/TLS Strong Encryption: An Introduction. *Apache.org* [online]. [cit. 2015-05-18]. Dostupné z URL: <http://httpd.apache.org/docs/2.2/ssl/ssl_intro.html>.

[7] ODVÁRKA, Petr. SSL protokol (1) - princip a přínosy. *Svět sítí* [online]. 2002 [cit. 2015-05-18]. Dostupné z URL: <http://www.svetsiti.cz/clanek.asp?cid=SSL-protokol-1-princip-a-prinosy-2542002>.

F DOKUMENTACE PRO VYUČUJÍCÍHO - LABORATORNÍ ÚLOHA - ČÁST 3.

F.1 Základní informace

V této úloze studenti realizují útok SSL strip. Tento útok je postavený na již známém útoku ARP spoofing z předchozí laboratorní úlohy. Realizace útoku probíhá na počítači - **útočník**, následně jsou generována data pomocí počítače **klienta**. Pomocí připravené webové stránky proběhne otestování útoku a zachycení odeslaného hesla klientem. Studenti by si měli uvědomit, že cílem útoku je cíleně obcházet využití zabezpečeného spojení HTTPS u klienta.

Ve druhé části jsou otestovány útoky na webovou aplikaci (Cross-sice scripting a SQL injection), kdy se studenti přihlásí do vytvořeného testovacího prostředí. V testovacím prostředí provádějí útoky zneužitím nešetřených vstupů (zde je třeba upozornit, že studenti musí pro úspěšný útok vkládat do formuláře správné znaky dle ASCII popsané v návodu).

F.2 Řešení úkolu

Vstupní kód pro zobrazení výplatní pásky daného uživatele (zde Pavel Kubový) bez znalosti hesla:

```
1' OR '1'='1' AND uzivJmeno='kubovypavel
```

F.3 Odpovědi na otázky

1. Odposlechem dat přenášených HTTPS přenosem není možné získat citlivé informace.
2. U ne-persistentního útoku je zákeřný kód proveden jednorázově, není uložen na serveru. U persistentního útoku je zákeřný kód uložen (např. v komentáři) a je proveden při každém načtení dané webové stránky.
3. Útočník odposlouchává HTTP komunikaci klienta. V případě nalezení odkazu na HTTPS v přenášených datech dojde k modifikaci tohoto odkazu na HTTP. Se serverem však útočník udržuje zabezpečenou HTTPS komunikaci.

G TEXT LABORATORNÍ ÚLOHY - ČÁST 4.

LABORATORNÍ ÚLOHA POČÍTAČOVÝCH ÚTOKŮ ČÁST 4.

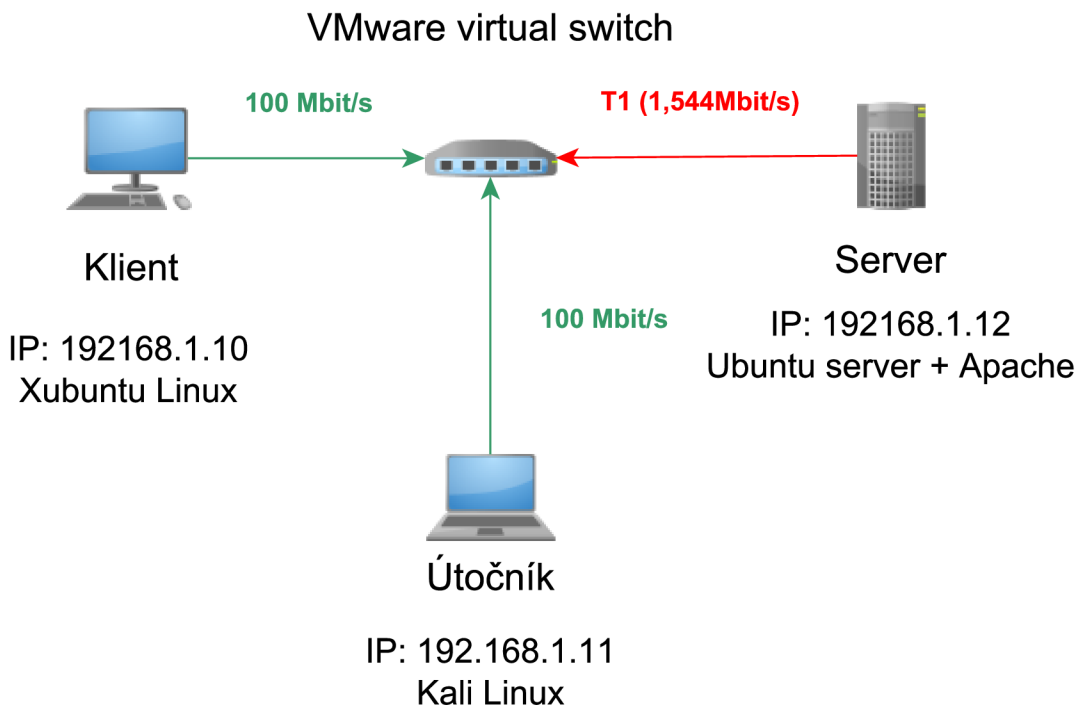
(Realizace DOS útoků a převzetí kontroly nad operačním systémem klienta)

Brno 2015

1 TEORETICKÝ ÚVOD

V této laboratorní úloze proběhne v rámci teorie seznámení s vybranými DOS útoky a proběhne seznámení s frameworkem Metasploit, který se využívá pro práci s exploity. V praktické části bude realizováno několik DOS útoků (ICMP flood, TCP flood, UDP flood a útok TCP reset). V další části proběhne realizace útoku na operační systém klienta s využitím chyby v nainstalované aplikaci. Úloha využívá níže uvedenou topologii 1.1.

1.1 Topologie sítě laboratorní úlohy

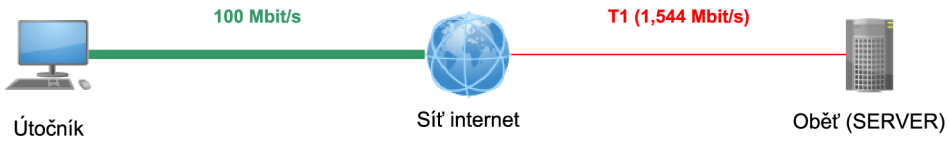


Obr. 1.1: Topologie sítě laboratorní úlohy

1.2 Útoky typu DoS (Denial of Service)

Dostupnost systému je nezbytná pro zajištění možnosti přístupu k datům. Útoky typu DoS (Denial of Service) útočí na systém za účelem omezení dostupnosti. Existující DoS útoky lze rozdělit na dvě základní skupiny.

DoS útoky na za účelem zahlcení komunikačního média: Tyto útoky se snaží vytížit linku připojeného systému, systém poté není schopný přijmout relevantní žádosti uživatelů a stává se tak nedostupným. Tyto útoky jsou také nazývány jako záplavové „floods“. Velký problém nastává, pokud má útočník připojení s větší šířkou pásma (např. 100 Mbit/s) než má server (např. 10 Mbit/s), jako je znázorněno na obr. 1.2. V dnešní době je pro tyto útoky využíváno převážně více počítačů, které mohou být rozložené v různých koutech sítě internet. Útok se poté nazývá distribuovaný DoS útok (Distributed Denial of Service).



Obr. 1.2: Rozdíl rychlostí v síťové infrastruktuře

Druhou skupinou jsou DoS útoky za účelem zhroucení cílového systému. V tomto případě je cíleno na hardware či software operačního systému. Na základě chyby v implementaci u OS je možné docílit zhroucení celého systému (z historie např. DoS útok Teardrop). Útočník může zaslat jeden nebo několik málo paketů, které jsou přesně nastaveny tak, aby cílový systém po jejich přijetí zkolaboval. Je však třeba zmínit, že i některé záplavové útoky mají schopnost docílit zhroucení systému.

ICMP záplava (ICMP Flood)

Útok využívající ICMP protokolu. Využívá pakety typu ICMP Echo a jsou využívány k zjištění dostupnosti cílového serveru.

Princip ICMP echa je takový, že útočník odešle ICMP Echo request na cílový počítač a server mu zpátky odešle ICMP echo reply. Je přitom zachována původní velikost paketu. Tímto způsobem je tedy linka serveru zahlcena hned dvojnásobně, jednak příchozím paketem a poté odesílaným paketem.

TCP záplava (TCP flood)

Jedná se o útoky založené na protokolu TCP. Existují záplavové útoky typu ACK, RST, FIN, NULL, URG, PSH, RST a podobné. Názvy výše uvedených útoků vychází z nastavení příznaků u TCP paketu.

1.3 Framework Metasploit

Metasploit Framework je nástroj určený pro penetrační testování. Je to vynikající nástroj pro vývoj, testování, správu a realizaci exploitů. Jedná se o Open Source nástroj a je k dispozici jak pro Unixové systémy, tak pro operační systém Windows (prostředí Cygwin).

Metasploit byl vytvořen v roce 2003, v prvopočátcích byl vnímám jako databáze publikovaných chyb a složil pro psaní nebo kombinování exploitů. V dnešní době se již jedná o plnohodnotně zdokumentované a aktualizované rozhraní, pomocí kterého je možné realizovat mnoho exploitů. Framework Metasploit obsahuje stovky použitelných modulů s exploity.

Práce s Metasploit frameworkem a realizace exploitu

V Unixovém prostředí se s frameworkem pracuje pomocí konzole, která se spouští z terminálu příkazem: `msfconsole`. Tímto je spuštěno prostředí metasploitu. Realizace exploitu probíhá pomocí následující procedury:

1. **Výběr exploitu:** V prvním kroku je vybrán a nakonfigurován modul s požadovaným exploitem. Exploitem je zde myšlen krátký kus programu, který má za úkol využít známé chyby v programu či operačním systému. Příkladem může být exploit, který využívá přetečení zásobníku pro přepsání návratové hodnoty a tím odkazuje na payload.

   Dostupné možnosti před výběrem exploitu:
   (a) **show exploit:** Získání seznamu dostupných exploitů.
   (b) **show exploit `nazev_exploitu`:** Zobrazení detailních informací o vybraném payloadu.
   (c) **use exploit `nazev_exploitu`:** Načtení vybraného payloadu.

   Dostupné možnosti při vybraném exploitu:

   (a) **show options:** Zobrazení parametrů, které jsou potřebné pro spuštění exploitu.
   (b) **show payloads:** Zobrazení seznamu payloadů, které jsou kompatibilní s vybraným exploitem.
   (c) **show targets:** Zobrazení seznamu systémů, proti kterým je možný exploit použít.
   (d) **show advanced:** Zobrazení dalších informací.
   (e) **show check:** Ověření zranitelnosti cílového systému.

2. **Výběr payloadu:** Dalším krokem je výběr a konfigurace payloadu. Payload je kód, který je spuštěn po úspěšném provedení exploitu. Pomocí payloadu je například možné řízení systému, je možné například otevřít porty na cílovém počítači a zajistit stažení souboru do cílového počítače nebo založit nový účet apod. Existuje více druhů payloadů, které se liší množstvím funkcí.
3. **Šifrování:** V dalším kroku je možné zvolit šifrovací techniku proti odhalení payloadu. Jedná se tedy o obcházení systému IPS (Intrusion-prevention System).
4. **Spuštění exploitu:** Posledním krokem je samotné spuštění exploitu.

2 PRAKTICKÁ ČÁST

V rámci praktické úlohy bude v první části demonstrováno několik útoků typu DOS. V druhé části proběhne praktické seznámení s frameworkem Metasploit a realizace jednoduchého útoku na operační systém klienta.

2.1 Realizace útoku - DOS

Spuštení virtuálních počítačů: K realizaci této úlohy je třeba virtuálních počítačů: **Útočník - Kali Linux 1.1.0**, **Server - Ubuntu Server 14.04.2 LTS** a **Klient - Xubuntu 14.04** Počítače se nacházejí ve složce **Bezpečnost**.

Nyní proběhne demonstrace útoku DOS za pomoci zahlcení linky serveru. Pro útok bude využit nástroj **hping3**.

ICMP Flood

Prvním z testovaných útoků bude ICMP flood útok. Jak již z názvu vyplývá, útok využívá protokolu ICMP (Internet Control Message Protocol). Tento protokol slouží převážně pro zjišťování dostupnosti služeb (např. PING), případně pro odesílání oznámení či chyb.

Klient, spuštění webového prohlížeče:
Nyní se spustí na klientském počítači webový prohlížeč.

Před samotným spuštěním útoku je třeba vymazat **cache paměť** prohlížeče klienta pro získání relevantních informací o době načítání webových stránek.

Klient, vymazání cache prohlížeče:
Stisknutím klávesové zkratky `CTRL + SHIFT + DELETE` při otevřeném webovém prohlížeči vyvolá nabídku **Clear All History**. Zde se zvolí volba **Time range to clear** na `Everything`, v části **Details** se ponechají všechna pole zaškrtnutá a potvrdí se **Clear Now**.

V tomto okamžiku je vhodné otestovat rychlost načítání stránek bez aktivního DOS útoku. Pro účely měření byl do webového prohlížeče nainstalován doplněk **app.telemetry Page Speed Monitor**. Tento doplněk měří dobu načítání každé

navštívené stránky založenou na metodě W3C navigation timing. Doplněk je již v této chvíli aktivní a lze ho nalézt v pravém dolním rohu okna webového prohlížeče, viz obr 2.1.

Obr. 2.1: Doplněk app.telemetry s informací o době načítání stránky

Nyní se přistoupí k samotnému testování rychlosti načtení stránky. Toto testování bude probíhat na stránce `www.test-domena.cz/dos`. Jedná se o klasickou webovou stránku, na které je umístěn relativně velký obrázek (793KB) pro vygenerování většího provozu.

Klient, načíst webovou stránku: Po úspěšném vymazání **cache** prohlížeče je třeba načíst webovou stránku `www.test-domena.cz/dos` a ve spodní části pravého okna zjistit čas načítání webové stránky.

Průměrná doba načítání stránky pro výše uvedenou topologii 1.1, kde je server záměrně připojen pomalou linkou T1 o rychlosti **1,544Mbit/s**, by měla být přibližně **4,5 s**.

V případě zájmu je možné toto měření provést vícekrát. **Je však nutné nejprve opět vymazat cache paměť** a poté stránku obnovit buď pomocí tlačítka **Obnovit** na stránce, případně klávesovou zkratkou **CTRL+R**.

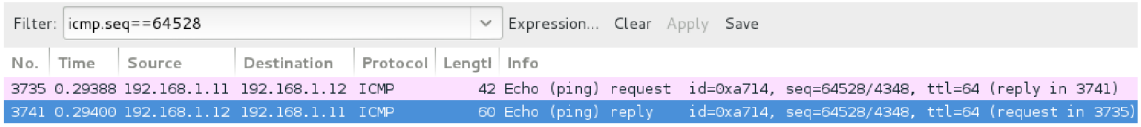
V tento okamžik se přistoupí k provedení DOS útoku. Využit k tomu bude již zmíněný nástroj **hping3**. Na počítači útočníka je třeba spustit okno terminálu a zadat níže uvedený příkaz:

Útočník, příkaz:

```
hping3 --icmp --flood 192.168.1.12
```

Kde příznak **icmp** znamená použití protokolu ICMP, příznak **flood** znamená odesílání rámců co nejvyšší možnou rychlostí a tím tedy i realizaci záplavy.

Po spuštění příkazu útočník generuje velké množství ICMP rámců typu **Echo request (ping)**. Jelikož je útočník připojen pomocí linky **100 Mbit/s**, tak jeho rychlost velmi převažuje oproti rychlosti linky serveru a ten je vzápětí zahlcen množstvím dotazů ICMP. Server následně odpovídá pomocí **Echo reply (ping)**. Na níže uvedeném obrázku 2.2 je zachycen průběh komunikace pro rámec s seq= 64528.



Obr. 2.2: Část zachycené komunikace při ICMP flood

Skutečnost, že je server opravdu zahlcen, je možné ověřit. Na severu je nainstalována jednoduchá CLI aplikace pro monitor síťového provozu **nload**. Nyní je tedy potřeba se přepnout na virtuální počítač serveru a přihlásit se

Server, přihlášení: Nyní je třeba se přihlásit na server s **uživatelským jménem:** `root` a **heslem:** `ubuntu`.

a spustit aplikaci **nload**

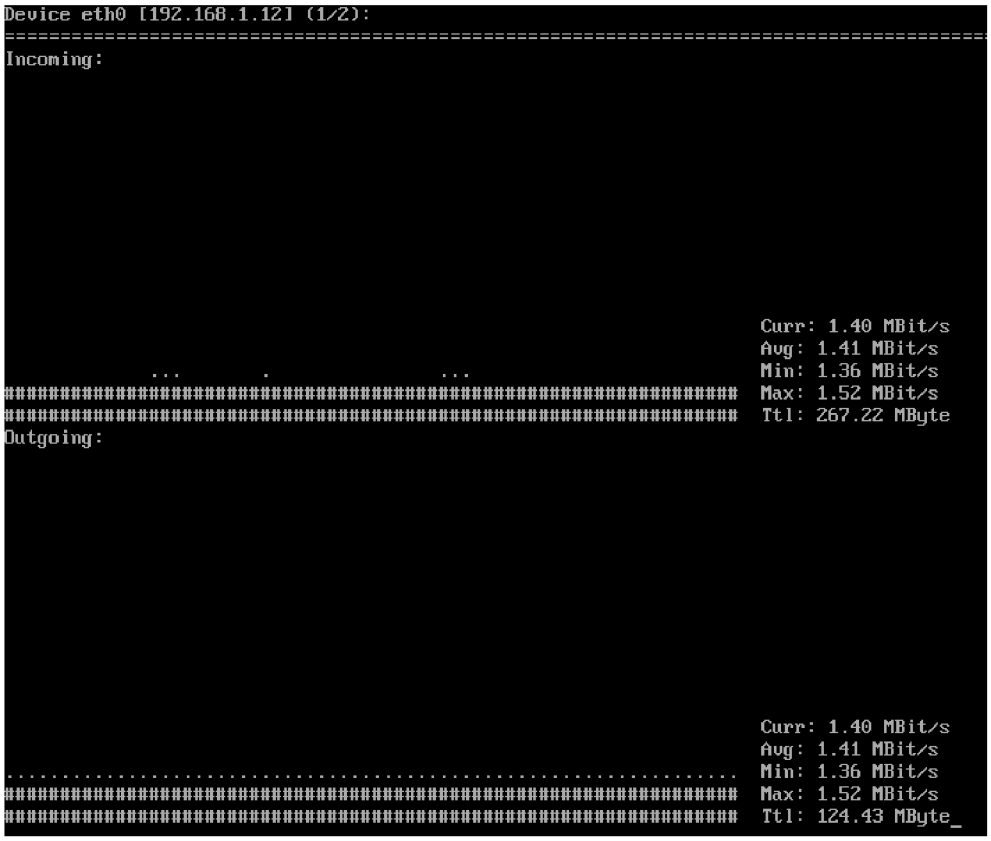
Server, příkaz: Nyní spustí aplikace **nload** pomocí příkazu: `nload`

Aplikace v momentě spuštění začíná měřit odchozí a příchozí tok dat v reálném čase. Rozhraní aplikace je velmi jednoduché. Ve vrchní části obrazovky je zobrazen příchozí tok dat **Incoming** ve spodní naopak odchozí tok **Outgoing**. Pomocí probíhajících znaků je zde snaha o vykreslení průběžného zatížení linky. Pro oba toky je možné sledovat aktuální tok dat (**Curr**), průměrný tok (**Avg**), minimální a maximální tok (**Min, Max**). V poslední řadě pak celkové množství přenesených dat od spuštění aplikace (**Ttl**). Popisované rozhraní je možné vidět na níže uvedeném obrázku 2.3.

V neposlední řadě je nutné ověřit, zda bude možné znovu načíst již dříve měřenou testovací DoS stránku. V tomto okamžiku je třeba ponechat program **hping3** spuštěný, znovu vymazat paměť prohlížeče a načíst stránku znovu.

Útočník, načtení testovací stránky: Nyní se opět provede test zobrazení webové stránky `http://www.test-domena.cz/dos`. Před opětovným obnovením stránky je nutné znovu vymazat paměť cache prohlížeče pomocí `CTRL+ SHIFT + DELETE`.

Na testované stránce nyní dojde k tak dramatickému zpomalení načítání stránky, že bude web na hranici dostupnosti. Při opakování pokusů může také selhat vytvoření spojení TCP a stránka se nenačte vůbec.



Obr. 2.3: Rozhraní programu **nload**

Pro získání obecnějších výsledků je nyní vhodné opakovaně provést proceduru vymazání **cache paměti** a **opětovného načtení** webové stránky. U toho je vhodné sledovat již zmiňovaný plugin **app.telemetry Page Speed Monitor**.

Po provedení několika načtení stránky je možné útok na počítači útočníka pozastavit.

> **Útočník, zastavení útoku:** Nyní se opět provede ukončení ICMP flooding útoku, zavřením okna terminálu případně klávesovou zkratkou `CTRL+C`

Při vytváření této úlohy byly pomocí zjednodušeného měření, při kterém byla načtena opakovaně webová stránka, získány následující výsledky:

Tab. 2.1: Měření času načítání webové stránky - ICMP flood

Číslo pokusu načtení stránky	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Doba načtení stránky [s] Běžný provoz	4,4	4,4	4,4	4,38	4,5	4,5	4,5	4,4	4,4	4,5
Doba načtení stránky [s] ICMP flood	31	31	N	215	43	42	N	N	25	N

Získané hodnoty by měly být obdobné i v případě této úlohy, z čehož lze usoudit že ICMP flood útok byl úspěšný a došlo k omezení dostupnosti webové stránky. Hodnota `N` v tabulce znamená, že stránku nebylo možné načíst.

TCP Flood

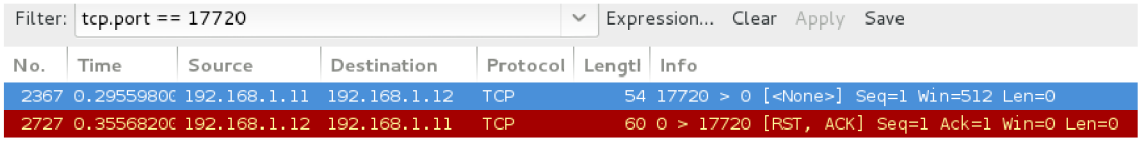
Nyní je možné přistoupit k dalšímu typu útoku, tím je TCP flood. Princip útoku je obdobný jako u ICMP flood, snahou je opět zahltit linku oběti. V případě TCP flood je odesíláno velké množství paketů TCP. Tyto pakety mohou mít různé příznaky jako `SYN`, `ACK`, `RST` apod... nebo nemusí mít příznak žádný.

Opět je využito programu **hping3**, tentokrát pouze s příznakem **flood** a cílovou IP adresou. Jelikož hping3 využívá jako defaultní protokol TCP, není nutné ho určovat. Níže je uvedený příkaz, který je nutné spustit pro realizaci útoku.

Útočník, příkaz:

```
hping3 --flood 192.168.1.12
```

V současné chvíli útočník generuje TCP pakety bez žádného příznaku v hlavičce paketu, čímž zahlcuje linku serveru. Jelikož server tyto pakety neočekával, zasílá zpět útočníkovi TCP paket s nastaveným příznakem RST, ACK. Server tedy linku vytěžuje i svými odpověďmi (stejně jako u ICMP floodu a odpovědí na PING request). Část zachycené komunikace pro paket se seq 17720 je zobrazena na obr. 2.4



Obr. 2.4: Část zachycené komunikace při TCP flood

Nyní je opět možné mapovat vytížení serveru pomocí programu **nload** a případně ověřit dostupnost serveru opětovným načítáním webové stránky pomocí webového prohlížeče.

Měření rozdílných časů načítání stránky nebylo realizováno, jelikož při tomto útoku nebylo možné ani v jenom z testovaných deseti opakování načíst webovou stránku. Při dalších opakování bylo docíleno načtení stránky po přibližně 5 minutách. Z tohoto pohledu se jeví TCP flood útok jako úspěšný.

UDP Flood

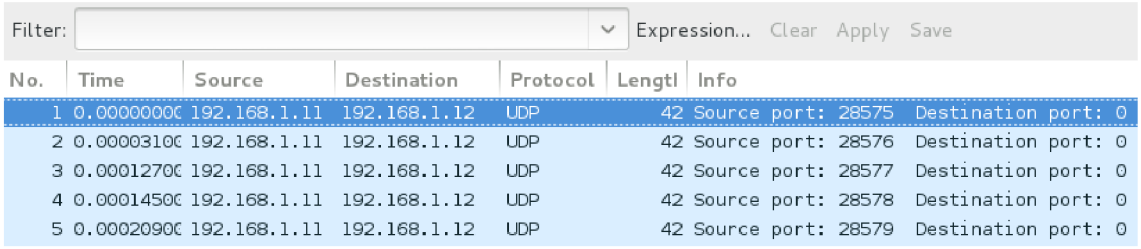
Posledním testovaným záplavovým útokem bude UDP flood. Princip je opět stejný jako v případě předchozích dvou útoků (ICMP flood a UDP flood). Při tomto útoku jsou generovány UDP datagramy.

Na cílovém serveru není otevřen žádný UDP port, ale to nebrání testu zahlcení linky pomocí UDP floodu. Útok je realizovaný níže uvedeným příkazem:

Útočník, příkaz:

```
hping3 --udp --flood 192.168.1.12
```

Nyní jsou generovány UDP datagramy útočníkem, příznak **udp** určuje protokol UDP a příznak **flood** generovaní paketů co nejvyšší rychlostí. Zachycené datagramy jsou zobrazeny na níže uvedeném obrázku 2.5.



Obr. 2.5: Zachycené UDP datagramy při UDP flood

Při prvotním testování útoku byla dostupnost cílové webové stránky relativně vysoká, pouze ve třech z deseti případů se stránka nenačetla. Ve většině případů byla průměrná doba načítání stránky 21,5 s viz tabulka 2.2.

Tab. 2.2: Měření času načítání webové stránky - UDP flood

Číslo pokusu načtení stránky	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Doba načtení stránky [s] Běžný provoz	4,4	4,4	4,4	4,38	4,5	4,5	4,5	4,4	4,4	4,5
Doba načtení stránky [s] UDP flood	13,6	N	N	N	16,8	40,8	21,8	22,1	23,9	11,7

Naměřené výsledky by měly být obdobné jako výše uvedené. Hodnota `N` v tabulce znamená, že stránku nebylo možné načíst.

Útok TCP reset

Při realizaci útoku DOS nemusí být vždy cílem zahlcení komunikace či zhroucení cílového systému. Typickým příkladem je DOS útok TCP reset.

V případě, že se útočník dostane mezi komunikaci klient/server (MITM), je také schopný tuto komunikaci přerušit.

Tento útok je možné použít v LAN sítích, kdy se útočník začne vydávat za cílový server (například pomocí již známého ARP spoofingu). Útočník však nemusí cílit na jediného klienta jako v předchozích úlohách, ale může falešnou informaci ARP šířit pomocí broadcastu do celé LAN sítě. Tímto je docíleno, že pokud se kterýkoli počítač v LAN bude chtít připojit k cílovému serveru, bude přesměrován na útočníka.

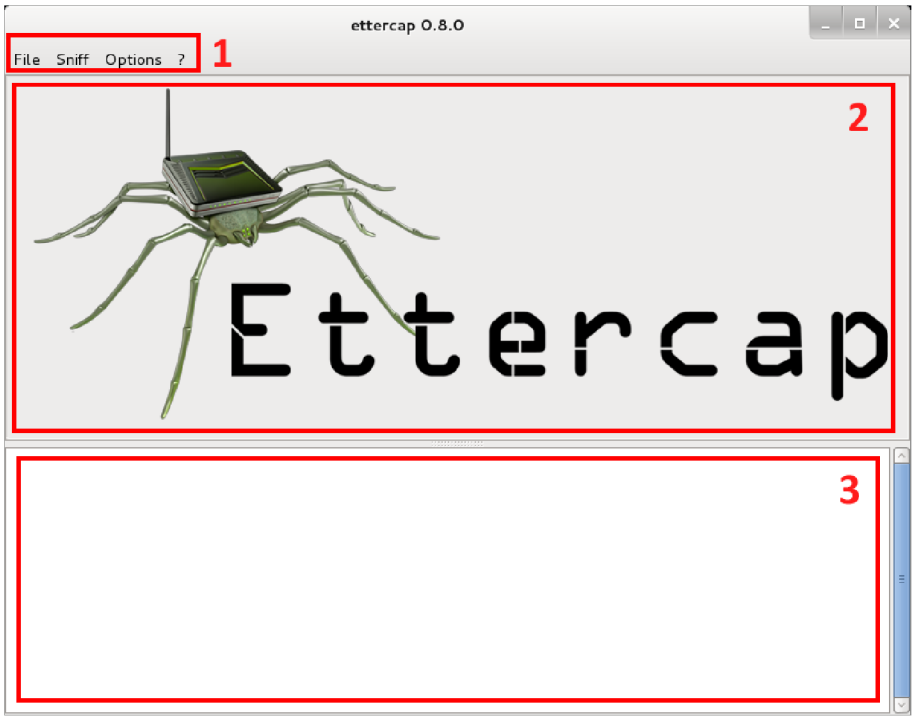
Pokud útočník zachytí takovýto paket určený pro cílový server, tak přečte sekvenční číslo paketu a odpoví klientovi paketem s příznakem RST (to znamená okamžitým ukončením spojení).

V první řadě je tedy nutné zajistit, aby všechen provoz procházel skrze útočníka. To bude realizováno za pomoci již zmiňovaného útoku ARP spoofing za pomoci programu **Ettercap**. Ettercap je program určený pro odposlouchávání sítě, realizaci útoků MitM, obsahuje také množství pluginů, které z něho dělají velmi zajímavý nástroj pro testování bezpečnosti.

Útočník, spuštění programu Ettercap: Nyní se spustí program Ettercap. Ten lze nalézt v hlavní nabídce **Applications/Kali Linux/Sniffing/Spoofing/Network Sniffers/Ettercap-Graphical**

Nyní se zobrazí základní prostředí programu Ettercap. Prostředí programu se skládá z hlavních tří částí, viz obr. 2.6. První částí (**1**) je hlavní nabídka, kde jsou vybírány typy útoků, jejich spuštění a ukončení. V druhé části okna (**2**) se zobrazují

záložky, ve kterých je možné provádět detailní nastavení. Spodní část (**3**) slouží pro zobrazování informačních hlášek apod.



Obr. 2.6: Rozhraní programu Ettercap

U Ettercapu je v první řadě nutné určit způsob odposlouchávání **Unified sniffing**. Unified sniffing odposlouchává komunikaci pomocí jednoho určeného rozhraní (v tomto případě eth0).

Útočník, volba způsobu odposlouchávání: V hlavní nabídce programu Ettercap je třeba zvolit způsob odposlouchávání Unified sniffing pomocí nabídky **Sniff/Unified Sniffing**, dále je třeba určit rozhraní, na kterém bude odposlouchávání probíhat. Zde je třeba potvrdit přednastavenou hodnotu **Network interface: eth0**.

V dalším kroku dojde ke spuštění odposlechu:

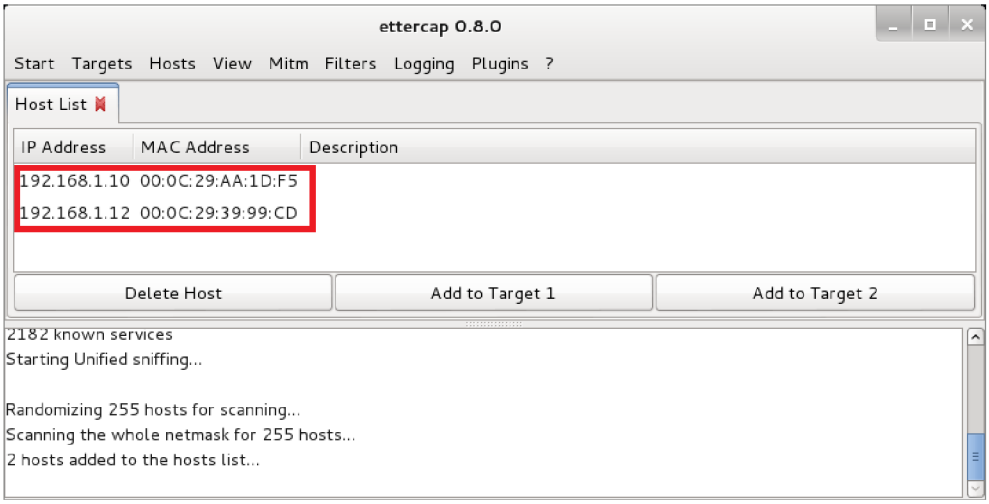
Útočník, spuštění odposlechu: V tomto kroku dojde ke spuštění odposlechu pomocí nabídky **Start/Start sniffing**

Ve spodní části Ettercapu je možné vidět informaci o spuštění odposlechu: `Starting Unified sniffing...`. Dále je třeba nalézt nalézt a určit cílový server, za který se bude útočník vydávat. Nejprve je nutné nalézt stanice v síti (hosts).

Útočník, nalezení stanic: Nyní dojde k nalezení dostupných stanic na síti, jejich IP adres a MAC adres. To se provede pomocí hlavní nabídky **Hosts/Scan for hosts**. Ettercap by měl nalézt dvě stanice, o kterých bude informovat ve spodní části Ettercapu `2 hosts added to the hosts list...`

Dalším krokem je zobrazení nalezených stanic a určení cílové stanice, na kterou bude probíhat útok ARP spoofing.

Útočník, výběr cílové stanice: Pro zobrazení nalezených stanic je třeba zvolit v nabídce **Hosts/Host list**. Nyní se zobrazí záložka Host List, ve které budou vidět IP adresy klienta a serveru viz obr 2.7.



Obr. 2.7: Nalezené stanice

Útočník, výběr cílové stanice: Výběr cílové stanice se provede označením dané IP adresy hosta a kliknutím na tlačítko **Add to Target 1**. Je tedy třeba vybrat IP adresu 192.168.1.12 (server). Po přidání IP adresy je opět zobrazena informace ve spodní části okna `Host 192.168.1.12 added to TARGET1`

Pozn: Zde je důležité poznamenat, že kdyby byl označen klient (192.168.1.10) a byl by přidán jako TARGET2, byl by vytvořen ARP spoofing stejný jako v předchozích úlohách. Jelikož toto není cílem, je zvolen pouze TARGET1 (192.168.1.12) a TARGET2 zůstává neurčen. Tímto je docíleno, že bude falešná informace ARP odesílána broadcastově do celé LAN sítě.

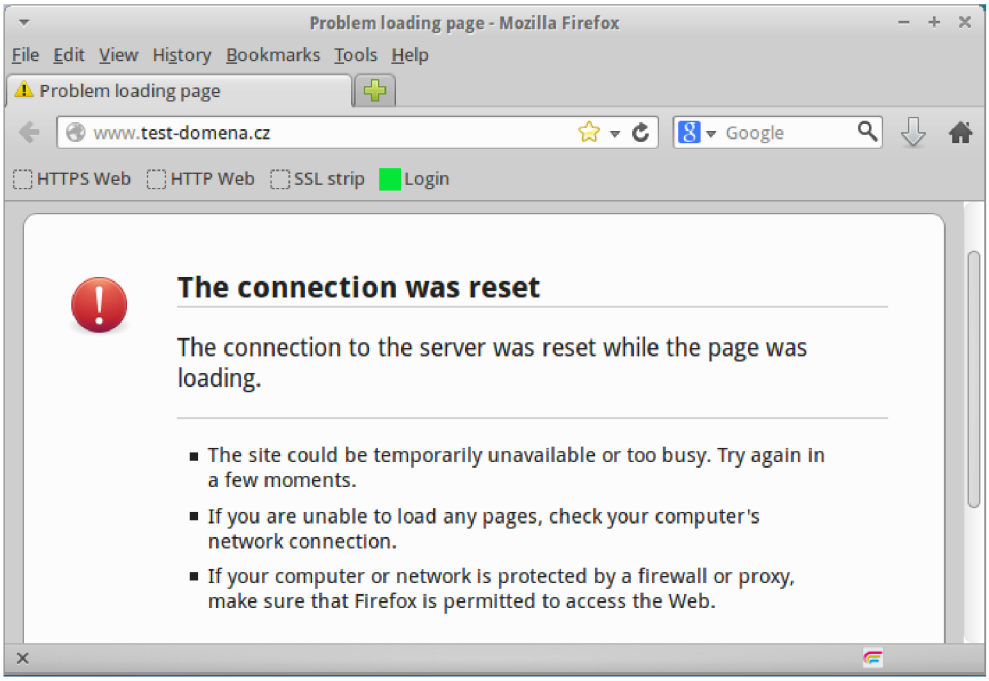
Útočník, spuštění ARP spoffingu: Nyní dojde ke spuštění ARP spoffingu pomocí nabídky **Mitm/ARP Poisoning**. Zde je třeba zaškrtnout pole **Sniff remote connections** a potvrdit.

Tímto je ARP spoffing kompletní. Každá stanice, která se bude kontaktovat server 192.168.1.12, bude přesměrována na útočníkovu stanici.

Nyní se přistoupí ke spuštění TCP reset útoku. K tomuto útoku bude využit pogram tcpkill, který bude zkoumat příchozí TCP spojení (od klientů na LAN síti), a v případě, že se bude klient připojovat na server 192.168.1.12, zašle již zmíněný paket s příznaky RST, ACK, a tím ukončí spojení. Nyní je třeba spustit terminál a níže uvedený příkaz:

Útočník: Nyní se sputí program tcpkill s určením cílového hota 192.168.1.12
příkaz: `tcpkill host 192.168.1.12`

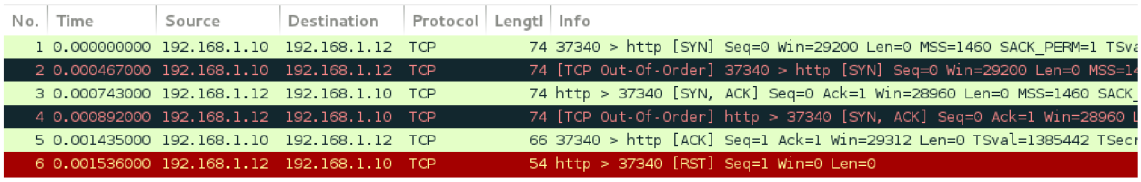
Útok je nyní aktivní. Pokud se klient pokusí připojit na `www.test-domena.cz`, tak je spojení okamžitě resetováno viz obr 2.8.



Obr. 2.8: Resetované spojení (úspěšný útok TCP reset)

Klient, otestování dostupnosti serveru: Nyní je třeba otevřít webový prohlížeč a zadat adresu `www.test-domena.cz`. Spojení by mělo být ihned resetováno obdobně jako na předchozím obrázku.

Odpovídající odchycená komunikace je na níže uvedeném obrázku 2.9. Klient se snaží vytvořit TCP spojení. V 6 odchyceném paketu však útočník odesílá RST paket.



No.	Time	Source	Destination	Protocol	Lengtl	Info
1	0.000000000	192.168.1.10	192.168.1.12	TCP		74 37340 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSva
2	0.000467000	192.168.1.10	192.168.1.12	TCP		74 [TCP Out-Of-Order] 37340 > http [SYN] Seq=0 Win=29200 Len=0 MSS=14
3	0.000743000	192.168.1.12	192.168.1.10	TCP		74 http > 37340 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_
4	0.000892000	192.168.1.12	192.168.1.10	TCP		74 [TCP Out-Of-Order] http > 37340 [SYN, ACK] Seq=0 Ack=1 Win=28960 L
5	0.001435000	192.168.1.10	192.168.1.12	TCP		66 37340 > http [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1385442 TSecr
6	0.001536000	192.168.1.12	192.168.1.10	TCP		54 http > 37340 [RST] Seq=1 Win=0 Len=0

Obr. 2.9: Zachycená komunikace při útoku TCP reset

Útok je tedy úspěšný. Je nutné připomenout, že útok bude úspěšný pouze v dané LAN síti. Server bude z ostatních částí internetu dostupný. Celkově lze útok hodnotit jako úspěšný.

Útočník: Nyní je třeba ukončit program **ettercap** a také ukončit program **tcpkill**.

2.2 Útok na operační systém s využitím chyby

Poslední část praktického cvičení bude věnována útoku na operační systém za účelem převzetí kontroly nad systémem. K tomuto účelu poslouží již popisovaný framework Metasploit, kdy bude zneužita slabina v programu Vsftpd verze 2.3.4.

Vsftpd (Very Secure FTP Daemon) je program, který slouží pro realizaci FTP serveru. Tento program byl vytvořen Chrisem Evansem a i v současnosti je hojně používaným programem pro vytvoření FTP serveru.

V roce 2011 však bylo zaútočeno na web hosting, na kterém byly umístěny instalační soubory programu vsftpd (v té době byla oficiální verze 2.3.4). Oficiální instalační soubor **vsftpd-2.3.4.tar.gz** byl nahrazen souborem, který obsahoval backdoor.

Soubor obsahoval následující slabinu: Pokud uživatel zadal při přihlašování na FTP server jako uživatelské jméno „smajlíka“ - `:)`, tak byl spuštěn příkazový řádek s právy root na portu 6200. Ačkoli se jedná o úsměvný útok, přesto jde o seriózní backdoor. Po tomto incidentu byl projekt Vsftpd přesunut na Google App Engine.

Prvním krokem je spuštění zranitelného Vsftpd serveru pomocí následujícího příkazu na straně klienta:

Klient, spuštění FTP serveru pomocí následujících příkazů:

```
sudo su
```

Pro spuštění příkazu bude požádováno **heslo**: `xubuntu` a následovně příkaz:

```
/usr/local/sbin/vsftpd &
```

Nyní je možné ověřit, zda FTP server běží na straně útočníka pomocí skeneru **nmap**, který se nachází deafultně v Kali Linux.

Útočník, skenování portů: Nyní se provede skenování otevřených portů na klientovi (192.168.1.10) pomocí **příkazu:** `nmap 192.168.1.10`.

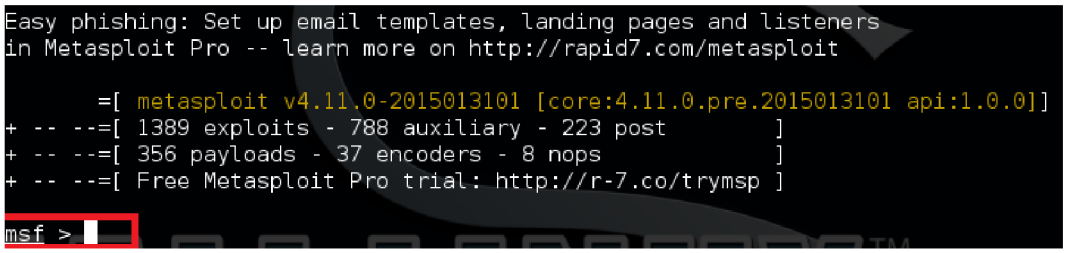
Po proskenování by měl být nalezen otevřený port 21 (FTP) obdobně jako na obrázku 2.10.

```
root@kali:~# nmap 192.168.1.10

Starting Nmap 6.47 ( http://nmap.org ) at 2015-05-04 05:42 EDT
Nmap scan report for 192.168.1.10
Host is up (0.00067s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
21/tcp open  ftp
MAC Address: 00:0C:29:AA:1D:F5 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.31 seconds
root@kali:~#
```

Obr. 2.10: Skenování portů pomocí nmap

Nyní se spustí samotný útok za pomoci již zmiňovaného Metasploit frameworku viz 1.3. Prvním krokem je spuštění metasploit konzole. Spuštění Metasploit konzole je náročnější proces, je tedy třeba trpělivosti:

Útočník, spuštění Metasploit konzole: `msfconsole` Načtení konzole bude indikováno interpretem `msf>` viz obr. 2.11.



Obr. 2.11: Konzola Metasploit frameworku

Po načtení frameworku se vybere odpovídající modul s exploitem. V tomto případě se jedná o **vsftps_234_backdoor**

Útočník, výběr exploitu:

```
use exploit/unix/ftp/vsftpd_234_backdoor
```

Úspěšně vybraný modul bude indikován změnou interpretu na:
`msf exploit(vsftpd_234_backdoor) >`

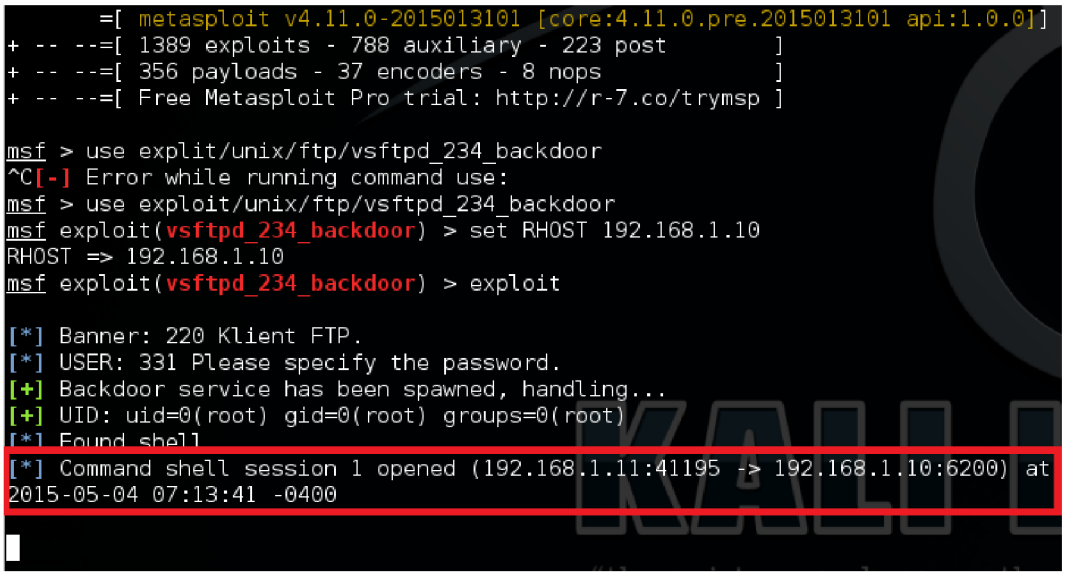
Nyní je zaveden modul, v současné době je možné zobrazit možnosti konfigurace modulu pomocí příkazu `show options`. Zde jsou zobrazeny dvě možnosti, které jsou vyžadované ke správnému spuštění modulu. Prvním je **RHOST**, který označuje IP adresu oběti. Dalším je **RPORT**, což označuje cílový port, na kterém je spuštěn zranitelný FTP server. Položka **RPORT** je defaultně nastavena na **21**, což odpovídá portu FTP serveru spuštěném na klientovi. Je však nutné zadat IP adresu klienta pomocí příkazu:

Útočník, nastavení IP adresy oběti: `set RHOST 192.168.1.10`

Tímto je potřebné nastavení modulu kompletní a je možné modul spustit pomocí níže uvedeného příkazu:

Útočník, spuštění Metasploit konzole: `exploit`

Po úspěšném útoku je navázáno spojení s klientem a je spuštěn vzdálený příkazový řádek, viz obrázek 2.12.



Obr. 2.12: Úspěšný exploit klientského počítače

Nyní je již možné zadávat vzdáleně příkazy na klientském počítači. Je možné například ověřit, pod jakým uživatelem je útočník připojen pomocí příkazu **whoami**: zde bude indikován `root`. Dále je možné ověřit informace o klientském počítači pomocí příkazu **uname -a**: zde bude indikován `Linux ubuntu 3.13.0`. Vypsat adresářovou strukturu pomocí příkazu **ls**.

2.3 Úkol

Úkolem je nyní vytvořit složku **vutbr** na ploše stanice klienta a ve složce vytvořit soubor **pwned** pouze pomocí vzdáleného připojení realizovaného výše popsaným exploitem. Jedná se o jednoduché příkazy, který se však nenachází v tomto návodu (pro vyhledání příkazů je doporučeno využít internet).

3 ZÁVĚR

Ve čtvrté laboratorní úloze byly realizovány DOS útoky. Útoky za účelem zahlcení linky oběti (serveru) byly v simulovaném prostředí úspěšné. Je však důležité říci, že linka serveru byla uměle snížena (1,544Mbit/s) oproti útočníkovi (100Mbit/s) za účelem snadné demonstrace útoku. V reálném prostředí jsou servery zpravidla připojeny rychlou linkou a jsou vybaveny systémy IDS, které znesnadňují tyto typy útoku. V reálném prostředí je ve většině případů realizovatelný pouze DDoS útok.

Druhá část laboratorní úlohy se zabývala frameworkem Metasploit a exploitem systému pomocí zranitelné aplikace nainstalované na klientském počítači. Útok byl pomocí frameworku úspěšný. Řešením je v tomto případě pravidelná instalace aktualizací, které opravují chyby v používaných aplikacích, a ověřování integrity instalovaných aplikací pomocí kontrolních součtů.

3.1 Kontrolní otázky

1. Jaký je princip DOS útoku TCP reset?
2. Jaký je rozdíl mezi DOS útoky záplavovými a DOS útoky využívající chyby?
3. Jakým způsobem by bylo možné odhalit, že je instalační soubor vsftpd podvržený útočníkem?

LITERATURA

[1] KOSEK, Jiří. Základy protokolu HTTP. *Kosek.cz* [online]. 1999 [cit. 2015-04-11]. Dostupné z URL: <http://www.kosek.cz/clanky/iweb/05.html>.

[2] Tcpkill(8) - Linux man page. *Die.net* [online]. [cit. 2015-05-18]. Dostupné z URL: <http://linux.die.net/man/8/tcpkill>.

[3] HALLER, Martin. Seriál Útoky typu DoS. *Lupa.cz* [online]. 2006 [cit. 2015-05-18]. Dostupné z URL: <http://www.lupa.cz/serialy/utoky-typu-dos/>.

[4] KÜMMEL, Roman. Metasploit Framework. *Soom.cz* [online]. 2006 [cit. 2015-05-18]. Dostupné z URL: <http://www.soom.cz/clanky/316--Metasploit-Framework>.

[5] Hping3(8) - Linux man page. *Die.net* [online]. [cit. 2015-05-18]. Dostupné z URL: <http://linux.die.net/man/8/hping3>.

H DOKUMENTACE PRO VYUČUJÍCÍHO - LABORATORNÍ ÚLOHA - ČÁST 4.

H.1 Základní informace

V této laboratorní úloze budou studenti realizovat útoky DOS. Principem prvních tří útoků bude zahlcení linky Serveru. Jedná se o útoky ICMP flood, TCP flood a UDP flood. Následně si studenti vyzkouší vytvoření ARP spoofing útoku pomocí programu Ettercap, na kterém následně realizují lokální útok TCP reset. Ve druhé části laboratorní úlohy proběhne spuštění FTP serveru s backdoorem na počítači klienta. Následně bude s použitím frameworku Metasploit realizován exploit. Studenti si zde ověří, že jsou připojeni na klientském počítači s právy uživatele root.

H.2 Řešení úkolu

Vytvoření složky **vutbr** na ploše a souboru **pwned** je možné následujícími příkazy:

```
cd Desktop
mkdir vutbr
touch vutbr/pwned
```

Pokud student vytvoří složku a soubor vzdáleně, budou vytvořené objekty na klientské stanici s právy **root** (signalizované ikonou „zámku“).

H.3 Odpovědi na otázky

1. Útočník monitoruje odchozí komunikaci klienta a zkoumá DNS dotazy. Pokud útočník nalezne DNS dotaz na doménové jméno, pro které má být podvržena jiná IP adresa, tak zasílá klientovi falešnou DNS odpověď.
2. Cílem záplavových útoků je primárně zahltit linku oběti (serveru). Cílem DOS útoků využívajících chybu je zaslat přesně nastavený paket, který způsobí pád cílového systému. Záplavový útok tedy využívá „hrubou sílu“, naopak k útoku využívající chybu je možné využít několika správně zacílených paketů.
3. Například zjištěním kontrolního součtu souboru pomocí hašovací funkce (např. sha1, md5) a následné porovnání s kontrolním součtem umístěným na důvěryhodných stránkách.

I OBSAH PŘILOŽENÉHO DVD

- Složka **Virtuální počítače** obsahuje komprimované soubory virtuálních počítačů (`Xubuntu - klient.zip, Kali-hacker.zip, Ubuntu-server.zip`).
- Složka **Laboratorní úlohy** obsahuje texty laboratorních úloh (`lab1.pdf, lab2.pdf, lab3.pdf, lab4.pdf`) a dále dokumentaci pro vyučujicího (`lab1_res.pdf, lab2_res.pdf, lab3_res.pdf, lab4_res.pdf`).
- Složka **Videonávody** obsahuje videonávody k lab. úlohám ve formátu mp4 (`lab1.mp4, lab2.mp4, lab3_SSLstrip.mp4, lab3_XSS.mp4, lab3_SQLinjection.mp4, lab4_backdoor.mp4, lab4_DOS.mp4`).
- Složka **Webová aplikace - server** obsahuje zdrojové kódy k webové aplikaci bežící na serveru.
- Složka **Webová aplikace - útočník** obsahuje zdrojové kódy webové aplikace umístěné na útočníkovi.
- Na DVD je přiložena elektronická verze diplomové pdáce (`DP-Plašil.pdf`).