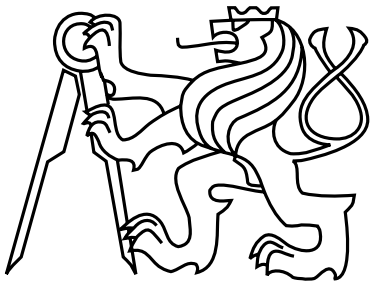
Na tomto místě bude oficiální zadání vaší práce

České vysoké učení technické v Praze
Fakulta Informačních Technologií
Katedra teoretické informatiky



Diplomová práce

Bezpečnostní studie aplikace

Jiří Fous

Vedoucí práce: Ing. Tomáš Zahradnický, EUR ING, Ph.D.

Studijní program: Informatika - magisterský

Obor: Systémové programování

5. května 2015

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Tomáši Zahradnickému, EUR ING, Ph.D., za vstřícný přístup, za informace ke zkoumané problematice a za asistenci při kompletaci textu. V neposlední řadě chci poděkovat svým blízkým, kteří mě po celou dobu podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.
Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze 5. 5. 2015 .....

Abstract

This work investigates the level of security of applications used by the general public and potential threats to the users in case of a possible attack, which abuses security vulnerabilities within these applications.

The aim of this work is to select an appropriate application according to the criteria and to perform subsequent security analysis of this application. The purpose of this analysis is to find security vulnerabilities like unexpected srashes of the application due to reaction on the user's input or weakly secured communication over the network. Found vulnerabilities in application have to be tested for their potential abuse with a simulated attack and in case of successful abuse some suitable countermeasures should be suggested.

Abstrakt

Tato práce se zabývá studiem úrovně zabezpečení aplikací používaných širokou veřejností a možnostmi ohrožení jejich uživatelů případným útokem, který využívá bezpečnostních zranitelností těmito aplikacemi způsobených.

Cílem této práce je vybrat vhodnou aplikaci na základě stanovených kritérií a tu následně podrobit bezpečnostní analýze. Účelem této analýzy je hledání bezpečnostních zranitelností v podobě neočekávaných pádů aplikace v souvislosti s reakcí na uživatelské vstupy a kontrola úrovně zabezpečení případné komunikace po síti. Nalezená potenciálně zranitelná místa aplikace jsou otestována na možnost jejich zneužití zkušebním útokem a v případě úspěšného útoku jsou následně navržena opatření a doporučení, která vedou ke snížení nebo předcházení rizikům ve spojení se zranitelností vzniklých.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V této práci se budeme zabývat bezpečnostní analýzou vybrané aplikace, která bude podrobena testům za účelem nalezení slabě a nedostatečně zabezpečených míst v aplikaci. Skutečný název aplikace je z bezpečnostních důvodů utajen, aby nebylo možné tuto práci zneužít jako návod pro provedení útoku na základě nalezených zranitelností, a je místo něj použit zástupný název `Kristýna`.

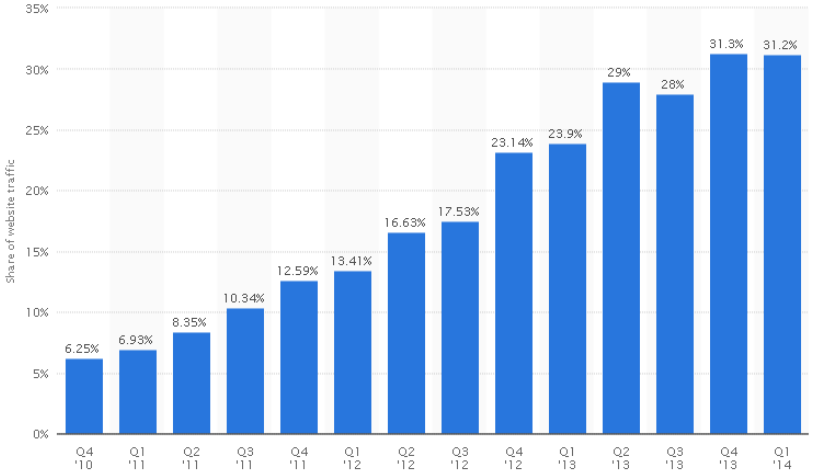
1.1 Motivace

1.1.1 Nárůst počtu zařízení připojených do sítě

Zejména v posledních letech s nástupem a bouřlivým rozvojem mobilních zařízení dochází k navyšování počtu zařízení připojených do sítě internet a rovněž k nárůstu doby, po kterou jsou daná zařízení v síti aktivní. Podle serveru Netmonitor.cz se počet mobilních uživatelů rapidně zvyšuje [6]. Tím vzniká větší prostor pro vývoj nového software, který těchto skutečností využívá, a to samozřejmě platí nejen na užitečný software, ale i na škodlivé programy, které mají mnohem větší možnosti se efektivně šířit, zaplavit co největší počet zařízení a poskytnout jejich vývojářům velkou výpočetní kapacitu zneužitelnou v mnoha směrech ať už je to různé rozesílání nevyžádané pošty, podnikání DoS/DDoS[1] útoků nebo shromažďování informací o uživateli.

Tento trend bude v budoucnu pravděpodobně pokračovat hlavně na poli mobilních zařízení, která stále ještě nabývají na významu zejména díky novým možnostem, které přicházejí spolu s velmi rychlým navyšováním výpočetního výkonu, nárůstem kapacit datových úložišť a přenosových linek mobilních operátorů poskytujících internetové připojení. Mobilní zařízení jsou navíc vybavována stále větším počtem různých senzorů, které rozšiřují jejich možnosti a poskytují aplikacím pro mobilní zařízení specifická data o poloze uživatele o přetížení a o dalších vlastnostech okolí a událostech v něm probíhajících. Skutečnost, že téměř každý člověk vlastní nebo brzy bude vlastnit a provozovat aspoň jedno z takových zařízení, a to i ve věkových skupinách lidí středního a pokročilého věku, které reagují na technologický vývoj pomaleji, představuje mimo velkého počtu výhod a nových možností hlavně nemalé bezpečnostní riziko pro soukromí celé společnosti. Tato bezpečnostní rizika lze zaváděním různých opatření omezit, ovšem vzhledem k množství obtížně ovlivnitelných

[1] Distributed Denial of Service - útok při kterém útočník znepřístupní službu ostatním uživatelům. Způsobů provedení je několik, nejčastěji je server záměrně přetížen nesmyslnými požadavky, případně je zahlcena síť, která v tomto důsledku přetížení nezvládne zajistit dostatečně rychlý a kvalitní přenos dat mezi serverem a uživateli [12].



Obrázek 1.1: Celosvětový vývoj počtu mobilních účtů[3]

faktorů, které snižují účinnost bezpečnostních opatření a tím míru rizika opět zvyšují, tato opatření ztrácí částečně nebo zcela svou účinnost a stačí jeden slabý článek v celém řetězu, aby došlo k úspěšné infiltraci útočníkem resp. jeho škodlivým kódem do cílového systému.

1.1.2 Lehkovážnost uživatelů

Tento faktor ovlivňující míru rizika je asi nejhorší, neboť největším nebezpečím pro sebedokonalejší systém bývá právě nezodpovědný uživatel, který svým jednáním může ať už vědomně či neúmyslně vytvořit bezpečnostní hrozbu, která nakonec umožní útočníkovi úspěšně napadnout koncové zařízení. Nejlépe lze ochránit systémy a zařízení ve firemních provozech, kde pracují z větší části pokročilí uživatelé se speciálními systémy, které vyžadují jistý stupěň odborné úrovně, a vzhledem k nutnosti jejich proškolení považuji tuto skupinu za nejméně nebezpečnou cílovou skupinu pro útok, protože na případné chyby vedoucí k podobným hrozbám, je obsluha předem upozorněna při instruktáži, aby se jich později vyvarovala a podobné systémy pokud neběží izolovaně od vnější sítě jako např. interní systémy pro řízení provozu výrobních linek a další speciální aplikace, mohou být zajímavým cílem pro napadení jen v případě, že útočník chce přímo poškodit daný provoz, protože jakýkoliv útok je až na výjimky motivován vidinou nějakého prospěchu či zisku a likvidace provozu výrobní linky uvedené jako příklad nebo podobných aplikací je poměrně speciální a úzce vyhraněná záležitost, která má význam jen pro malou skupinu lidí.

Protipólem jsou zařízení, která jsou používána širokou veřejností. Narozdíl od firemních specializovaných systémů s odbornou obsluhou, která čítá omezenou skupinu proškolených lidí pracujících na profesionálně udržovaných systémech, před sebou máme skupinu běžných naivních uživatelů, kteří z většiny budou patřit mezi naprosté laiky bez povědomí o jakýchkoliv zásadách bezpečného používání svých zařízení. Zaměření takových cílových skupin má hned několik na první pohled nesporných výhod. První dvě nejvýznamnější jsou „počet potenciálních cílů“ a „lehkovážnost uživatelů“. Nezodpovědní uživatelé obvykle otevřou útočníkovi pomyslné dveře do systému sami a dobrovolně a stačí k tomu obvykle jen poslat nějakou zajímavou zprávu na jejich e-mail, sociální síť a další možné komunikační kanály, která je zaujme natolik, aby provedli nějakou požadovanou akci

— typicky klikli na odkaz, který je zavede na nakaženou stránku, kde dojde skrytou formou k infekci systému nebo aby spustili přílohu, která se postará o to samé. Další možností je nalézt bezpečnostní díru v samotném systému a následně takovou bezpečnostní díru zneužít. V takovém případě dokonce nemusí být ani vyžadována nějaká akce od uživatele a útok může proběhnout zcela bez jeho účasti. V obou případech celý efekt posiluje první zmíněná výhoda a to je počet uživatelů, kteří tvoří cíle útoku, protože při útoku ve formě phishingu[2] znamená více napadených uživatelů logicky i větší počet úspěšně podvedených jedinců, který samozřejmě v absolutních číslech roste s kvalitou provedení útoku, stejně tak při zneužití objevené zranitelnosti v masově používaném software nebo i přímo v operačním systému, je větší počet cílů zárukou vlivu nad více zařízeními.

1.1.3 Proč jsou lehkovážní problémem i pro zodpovědné

Nezodpovědní uživatelé díky své lehkomyslnosti a hlavně neznalosti možných hrozeb a způsobů, jak jim předcházet, dávají útočníkům svá zařízení k dispozici. To je problémem i pro další uživatele a to si většina lidí vůbec neuvědomuje. Zařízení, nad kterým má útočník plnou kontrolu, totiž může být jeho prodlouženou rukou v lokální síti, do které je připojeno. Provádění útoku na zařízení ze stejné podsítě je totiž mnohem jednodušší, než na něj útočit „z venku“ už třeba kvůli nastavení firewallu, které pro lokální podsíť bývá méně striktní než pro vnější síť. Důvodů je samozřejmě více. Za zmínku stojí například možnost útoku typu DNS spoofing[3], lepší možnosti při zjišťování struktury vnitřní sítě a také mnohem lepší anonymita útočníka, protože je schopen zajistit, aby stopa po případném dalším útoku skončila právě u napadeného zařízení a jeho nic netušícího uživatele.

1.1.4 Důvody vedoucí k neopatrnosti

Důvodů, které uživatele vedou k neopatrnosti je hodně, ale asi nejvýznamnějšími jsou pohodlnost a lenost uživatelů. Za těmito dvěma hlavními následuje zvědavost. Pohodlnost se projevuje zejména u často prováděných akcí. Typickým příkladem jsou přihlašovací údaje. V dnešní době má každý běžný člověk i desítky různých uživatelských účtů v desítkách systémů. Lidé ovšem při zohledňování svého pohodlí často neodlišují rozsáhlost možného dopadu na jejich soukromí při prolomení zabezpečení do jejich účtu na internetové diskuzi o vaření a při stejném problému v případě účtu k firemní databázi klientů nebo k internetovému bankovnictví. Dopad v prvním případě je téměř nulový, ale v dalších dvou případech může dojít k závažnému narušení soukromí a i vzniku nemalých hmotných škod a k porušení zákonů např. o ochraně osobních údajů v případě zcizení databáze klientů. Velice často se i dnes stále setkávám s lidmi, kteří mají své osobní heslo stejné na všech účtech od e-mailu přes sociální sítě až po bezvýznamné internetové diskuze a snad jediní, kteří dbají doporučení vyvarovat se takového jednání, jsou ti, kterým byl v minulosti už nějaký z účtů odcizen a ve spojení s tím došlo k nemalým problémům nebo dokonce k páchání trestné činnosti. Podobné chování se bohužel netýká zdaleka jen volby hesel. Přitom řešení, které minimalizuje nebezpečí, nebývá nijak extrémně složité a dokonce se ani nutně nemusí jednat o snížení komfortu. V případě hesel stačí, aby si uživatel své účty roztřídil dle důležitosti. Na kritické aplikace je potřeba zvolit

[2] V překladu „rhybaření“ - je forma útoku, která zneužívá na základě promyšlených psychologických postupů uživatelovy důvěry a přiměje ho k zadání svých citlivých údajů do podvrženého formuláře, který je následně odeslán útočníkovi [13].

[3] Otrávení cache, která uchovává informace o DNS zaznamech a tím je umožněno přesměrovat uživatele na podvodnou stránku [13].

zvláštní a dostatečně dlouhé heslo, ale pro účty na různých webových diskuzích a podobných službách stačí zvolit slabší heslo, které lze použít i vícekrát. Kritických aplikací bude jistě podstatně méně, než těch, kde není nutné tolik dbát na sílu a jedinečnost hesla. Typicky se jedná o internetové bankovnictví a e-mail, kde je potřeba být nejopatrnější.

Pro plošný útok však bývá častější výhodou pro útočníka zvědavost uživatele. Kdo by se nechtěl podívat na zajímavé či snad přímo skandální video, fotografii nebo si nestáhl zdarma nejnovější filmový trhák, který se právě uvádí do kin? Dokonce i když uživatel tuší, že to možná bude podvrh, pro jistotu to obvykle vyzkouší, aby se ujistil, že o nic nepřichází a na základě tohoto vzorce chování lze velice jednoduše uspět u velké části takových uživatelů. Ti opatrnější mají ovšem výhodu v tom, že v našem modelovém příkladě s hesly, přijdou obykle jen o jeden účet, který navíc později zřejmě dokážou obnovit a získat zpět prostřednictvím funkcí jako je „obnova hesla“ pomocí e-mailu, SMS apod.

1.1.5 Útok na nejslabší článek v řetězu

Poměrně aktuálním tématem jsou útoky na tzv. chytré telefony, které v posledních letech zaplavily trh s mobilními telefony. Cílem útoků jsou zejména proto, že jejich prostřednictvím uživatelé ovládají svá bankovní konta, ke kterým mají zřízenou službu internetového bankovnictví. Cílem útočníka je proto získat kód z potvrzovacích SMS zpráv, které na mobilní telefon uživateli posílá systém z banky. Zde se může projevit další slabina v myšlení uživatelů, kteří svůj mobilní telefon patřičně střeží, svůj notebook rovněž, ale stolní počítač u nich v domě již není předmětem zájmu, protože s ním pracují obvykle pouze ostatní členové rodiny a pro přístup do banky se stejně tento stroj nepoužívá. Takto uvažující uživatel ovšem zapomněl, že po příchodu domů se jeho dvě střežená zařízení nacházejí na stejné podsíti a v případě, že stolní počítač už je nakažen, je mnohem jednodušší zaútočit a úspěšně ovládnout další zařízení na síti. To samé platí pro špatně zabezpečené sítě ve školách, podnicích, případě v kavárnách, hotelech, dopravních prostředcích a dalších „free wifi“ sítích.

1.1.6 Vývoj aplikací bez důrazu na bezpečnost

Zranitelnost, která umožní útočníkovi zmocnit se postiženého systému, vznikne při programování software tím, že vývojář udělá nějakou chybu. Taková chyba může být různé povahy, od banálních začátečnických chyb, které spočívají v chybném řízení cyklů a plnění paměťových bufferů a které pak vedou na zranitelnosti typu „buffer overflow“[4], až po chyby, které vzniknou nesprávným nastavením přístupových práv k definovaným zdrojům v programu a dalšími pochybeními při psaní kódu.

1.1.7 Secure by design

Bezpečnost a její zohlednění má často podobnou důležitost při řízení projektu jako testování. Obojí se na začátku jeví jako nepotřebné a zbytečné, neboť hlavní důraz se klade na funkčnost, ale zapomíná se na to, že zejména zabezpečení je nutné promýšlet na samém začátku vývoje, protože

[4] Jedná se o zranitelnost, která vznikne chybou v programu, která umožňuje zapsat za konec vyhrazené paměti. To může např na zásobníku způsobit, že útočník zadáním vhodného vstupu přepíše data na zásobníku a změnou návratových adres se mu může podařit spustit jeho škodlivý kód [25].

po dokončení softwarového produktu je už pozdě řešit bezpečnostní problémy a obalovat výsledný programový kód narychlo implementovaným zabezpečením kritických oblastí. To paradoxně může situaci ještě zhoršit. Tato situace je ale dána zejména nároky na co nejnižší cenu software a proto se na bezpečnost dbá poctivě jen tam, kde to vyžaduje buď zákon nebo interní nařízení v rámci firemního prostředí a někdy i tam úroveň zabezpečení pokulhává[5]. Ovšem zejména software, který je vyvíjen menšími společnostmi, drobnými podnikateli a nadšenci, bývá často zranitelný právě z výše zmíněných důvodů a to jsou nedostatek financí a nezkušenost vývojářů, kteří mnohdy nedbají ani základních pravidel, aby předcházeli při vývoji vzniku hrozeb.

1.1.8 Způsob šíření škodlivého SW

Nejdůležitější pro škodlivý software je zvládnutí způsobu jeho efektivního šíření. Dřívější způsoby šíření přes vyměnitelná média se stále používají, zejména to platí pro flash paměti, které jsou dnes hojně využívány pro výměnu souborů mezi počítači, které nejsou připojené do sítě, a také je lidé používají např. v kopírovacích centrech, kam na flashdisku donesou svou práci, a kromě výtisku si domů odnesou zpět i důmyslný software, který se po připojení flashdisku infiltruje do hostitelského systému. Tento způsob šíření je ale oproti šíření po síti internet mnohonásobně pomalejší. Dnešní prostředí, kdy většina zařízení je neustále připojených do sítě internet, je naprosto ideální živnou půdou pro tento druh kriminality, protože rozšíření nového škodlivého programu, na který navíc ještě nereagují ani antivirové programy, a nakažení velkého množství počítačů může být otázkou i hodin nebo několika málo dní. Záleží jen na důmyslnosti metody šíření.

1.2 Specifikace cíle

1.2.1 Cíl práce

Cílem práce bylo stanoveno najít nějaký zajímavý a co nejoblíbenější tzn. pravděpodobně i nejpoužívanější program mezi uživateli v České republice, který není s otevřeným zdrojovým kódem, ale je volně dostupný na internetu a jeho vydavatel patří mezi méně známé či úplně neznámé vydavatele nebo vývojáře software. Takový zástupce bude po tomto „výběrovém řízení“ podroben detailnější analýze z hlediska bezpečnosti od jeho instalace do systému až po jeho spuštění.

Úkolem je nalézt zranitelnosti, které by mohly ohrozit uživatele programu a způsobit mu při jejich zneužití škody v systému případně i uživateli, pokud by došlo k ohrožení jeho osobních dat, přístupových údajů do kritických systémů nebo třeba i k odcizení citlivých informací z lokální databáze, která může obsahovat údaje třetích stran jako jsou databáze zákazníků a jiné důvěrné materiály. Takové zranitelnosti budou prověřeny z hlediska jejich zneužitelnosti pro útok. Po identifikaci případných zranitelností budou uvedeny i postupy a instrukce pro jejich opravu, zásady pro předcházení takovým chybám již při vývoji, případně i doporučení pro uživatele, jak i při používání takového zranitelného programu minimalizovat rizika vzniku incidentu, pokud je to možné.

[5] Například Opencard a slabé šifrování karet, které prolomil německý inženýr Timo Kasper [14], nebo pochybení při správě certifikátů v datových schránkách, kde došlo k použití certifikátů po konci jejich platnosti [20].

Kapitola 2

Analýza

V této kapitole se budeme zabývat stanovením kritérií a postupu pro výběr aplikace k testování, určením vhodných metod pro systematické hledání zranitelností a výběrem pomocného programového vybavení, které bude potřeba k realizaci stanovených úkolů.

2.1 Stanovení požadavků na výběr programu

Na základě zadání práce vytvoříme seznam vhodných kritérií, podle kterých následně provedeme výběr aplikace pro bezpečnostní analýzu.

2.1.1 Kritéria výběru cílové skupiny software pro analýzu

Počítačových programů je obrovské množství. Základní myšlenkou při výběru bylo zacílit na takový program, který by představoval největší riziko pro bezpečnost uživatelů a takový program by měl splňovat několik základních kritérií.

1. **Oblíbenost u uživatelů**: Čím oblíbenější program je, tím více uživatelů ho používá a tím rozsáhlejší dopad může mít případná zranitelnost. Proto se v této práci zaměřím zejména na software poskytující služby pro širokou veřejnost. S tímto kritériem souvisí i následný výběr operačního systému v další kapitole, na kterém budou testy programů probíhat.

2. **Původ programu**: Jako nezajímavým se jeví software od renomovaných společností, který má dlouholetou tradici a který je vyvíjen předními světovými odborníky, kteří v rámci udržení dobré pověsti společnosti mají většinou zájem na tom udržet její jméno neposkvrněné nějakým skandálem v souvislosti se špatně zabezpečeným produktem, naproti tomu stojí velké množství drobných ale i rozsáhlejších programů, které jsou vyvíjeny různými neznámými společnostmi z celého světa nebo i jednotlivými nadšenci a mnohdy jsou uvolněny jako freeware případně shareware. Takový software bývá velice snadno dostupný na různých serverech, které fungují jako agregátory podobných produktů a nabízejí je zdarma ke stažení svým návštěvníkům. Takový software není obvykle precizně zkontrolován na přítomnost škodlivého kódu a i přesto, že některé servery zabývající se shromažďováním takových programů a jejich následnou distribucí tvrdí, že veškerý software na jejich serveru je bezpečný, protože byl

zkontrolován některým z antivirových programů, kterému server pravděpodobně dělá tímto způsobem reklamu, nelze se na takové tvrzení absolutně spolehnout.

3. **Dostupnost zdrojových kódů programu**: Program, jehož zdrojové kódy jsou veřejně přístupné, je možné poměrně snadno zkontrolovat na přítomnost škodlivého kódu a na případné zranitelnosti uvnitř programu, které by mohly vytvořit zneužitelnou bezpečnostní díru po instalaci do systému uživatele. Prověření programů, které nemají veřejně přístupné zdrojové kódy, je řádově složitější, protože k jejich analýze je potřeba mnohem více úsilí, zkušeností a speciálního softwarového vybavení, které umožňuje sledovat práci programu uvnitř hostitelského operačního systému a poskytnout tak analytikovi potřebné informace o chování zkoumaného programu, aby byl schopen z nasbíraných dat vyvodit závěry o jeho vlastnostech a činnostech v tomto případě hlavně z hlediska bezpečnosti jeho provozování.

2.2 Výběr testovacího prostředí

Pro zajištění testování jednotlivých programů bude potřeba pečlivě stanovit požadavky na testovací prostředí. Je potřeba u testování zajistit srovnatelné počáteční podmínky pro každý testovaný program s důrazem zejména na poskytnutí vždy stejného výchozího stavu systému pro zkušební instalaci programu. Rovněž je nutné zohlednit časovou náročnost zajištění těchto výchozích podmínek vzhledem k počtu testovaných instalací a zjevné potřebě mít možnost testy jednoduše opakovat pro porovnání a ověření zaznamenaných výsledků.

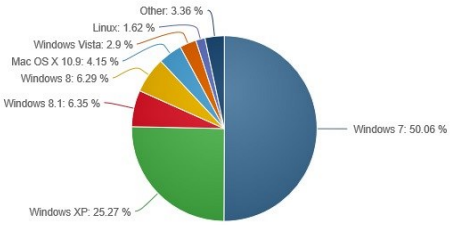
2.2.1 Výběr operačního systému

Při výběru operačního systému bude potřeba zohlednit zejména první kritérium pro stanovení cílů práce a to je použití širokou veřejností. Z veřejně známých a používaných operačních systémů jako OS X, Linux a Windows je u nás v ČR jednoznačně nejčastěji se vyskytujícím operační systém od společnosti Microsoft, Corp. s názvem Microsoft Windows. Windows se aktuálně vyskytuje v několika verzích, proto jsme z českého serveru idnes.cz získali statistiku o podílech jednotlivých verzí operačních systémů na trhu, abychom mohli na základě těchto údajů vybrat nejpoužívanější a nejrozšířenější verzi.

Z této statistiky vyplývá, že nejpoužívanějšími systémy jsou starší Windows XP a s nepatrně více než 50% vede Windows 7. Alternativní systémy jako Linux a jeho varianty nebo OS X jsou součástí menšiny používaných operačních systémů, a proto se můj výběr soustředil hlavně na verze OS Windows. Vzhledem k ukončené podpoře a zastaralosti operačního systému Windows XP, jsme zvolili operační systém Windows 7.

2.2.2 Využití nástrojů pro návrat systému do počátečního stavu

Využití standardního fyzického počítače spolu s čistou instalací nějakého operačního systému s jeho počátečním nastavením a případným doinstalováním ovladačů hardware a periferií, se samozřejmě na první pohled jeví jako značně neefektivní řešení, které v sobě skrývá vzhledem ke své složitosti poměrně velké časové náklady, které by pravděpodobně značně převýšily časovou náročnost



OPERATING SYSTEM	TOTAL MARKET SHARE
Windows 7	50.06%
Windows XP	25.27%
Windows 8.1	6.35%
Windows 8	6.29%
Mac OS X 10.9	4.15%
Windows Vista	2.90%
Linux	1.62%
Mac OS X 10.6	1.11%
Mac OS X 10.8	0.97%
Mac OS X 10.7	0.87%
Mac OS X 10.5	0.22%
Windows NT	0.08%
Mac OS X 10.4	0.05%
Windows 2000	0.03%
Mac OS X (no version reported)	0.01%

Obrázek 2.1: Podíl operačních sytémů na trhu [23].

samotných testů při nutnosti po každém testu a sesbírání potřebných údajů provést odstranění nainstalovaného sytému a celý proces jeho opětovné instalace na zformátovaný pevný disk opakovat. Z tohoto důvodu jsem hledal jiná a efektivnější řešení, která by výše zmíněné nevýhody neměla.

Systém Windows disponuje funkcí s názvem *obnovení systému*, která umožňuje provádět ukládání stavu systému, registrů a důležitých systémových souborů bez ovlivnění osobních souborů uživatele jako jsou dokumenty, multimédia a další soubory, které nejsou součástí operačního systému [16]. Využití tohoto nástroje pro účely této práce a pro zajištění návratu systému do výchozího stavu při testování jednotlivých programů jsem zvažoval, ale vzhledem k tomu, že bod obnovení obsahuje informace pouze o systémových souborech a registru, nemusela by tak být zajištěna podmínka pro testování, která udává, že každá jednotlivá instalace musí být prováděna ze stejného výchozího stavu celého systému, proto by bylo potřeba nejdříve získat seznam souborů ovlivněných instalací testovaného programu a následně ho porovnat se seznamem zálohovaných souborů nástrojem pro obnovení systému, zda-li skutečně po návratu k výchozímu uloženému stavu bude splněna podmínka, že celý systém bude ve stejném stavu jako před instalací. Navíc návrat systému do předchozího uloženého stavu trvá v závislosti na provedených změnách také poměrně dlouhou dobu a je potřeba celý stroj restartovat. Z těchto dvou důvodů, kterými je nutnost ověření funkčnosti tohoto nástroje a nezanedbatelná časová prodleva při provádění obnovy systému jsem tuto možnost považoval pouze za náhradní řešení v případě, že by se nepodařilo nalézt řešení lepší.

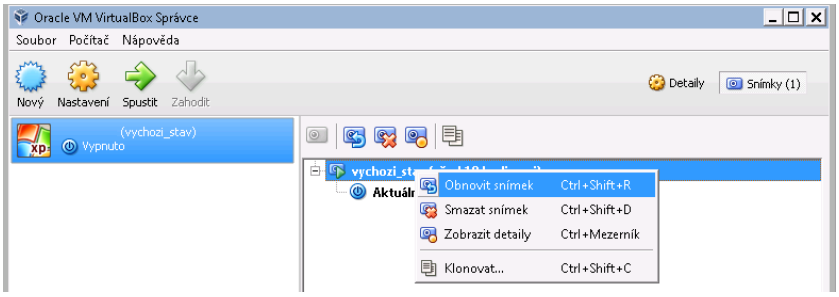
2.2.3 Použití virtuálního stroje a jeho výhody

Jako zajímavým nápadem se jevilo využití některého virtualizačního nástroje, kdy by odpadla nutnost připravovat fyzický stroj pro testování, ale stačilo by takový stroj pouze simulovat prostřednictvím tohoto software. Nástrojů pro virtualizaci OS je více. Zkušenosti jsem měl doposud zejména

s nástrojem `Virtual Box` od společnosti `Oracle`. Při výběru virtualizačního nástroje jsem zohledňoval hlavně podporu instalovaného operačního systému vybraného pro testování, kterým je Windows 7, a možnost návratu systému do uloženého stavu resp. zálohování vytvořeného virtuálního počítače a jeho případné duplikování. Podpora pro Windows 7 se ve Virtual Boxu nachází a samotné zálohování virtuálního stroje je možné poměrně jednoduchým způsobem, a to je duplikací virtuálního pevného disku, který lze následně připojit jako pevný disk nového virtuálního stroje.

2.2.4 Uložení stavu virtuálního stroje

Po ověření funkčnosti virtualizačního nástroje Virtual Box potřebné pro účely mé práce jsem prozkoumal i dodatečnou funkci, která umožňuje vytvářet tzv. snímky virtuálního stroje. Tato funkce umožňovala ve kterémkoliv okamžiku pozastavit a uložit stav virtuálního stroje, který může být kdykoliv velice jednoduše obnoven a to bez ohledu na právě spuštěné procesy a probíhající úlohy v systému. Takto obnovený systém z předem uloženého stavu lze použít pro zajištění požadovaného stejného výchozího stavu systému pro testování jednotlivých programů.



Obrázek 2.2: Snímek virtuálního stroje v aplikaci Virtual Box

2.2.5 Pozastavení virtuálního stroje

Virtuální stroj je možné kdykoliv pozastavit a následně jedním příkazem opět rozběhnout. Během doby, kdy systém neběží, zůstává stále nahraný v operační paměti hostitelského počítače, ale nespotřebovává žádný jeho procesorový čas a aplikace v něm spuštěné jsou pozastaveny. Tato funkce je užitečná zejména ve chvílích, kdy je vhodné nebo dokonce nutné pozastavit prováděné procedury a například zjistit nebo ověřit nově získané informace předtím, než lze opět pokračovat v práci na virtuálním stroji. V případě, že pozorujeme činnost nějakého programu, kterou se snažíme zachytit a analyzovat, poskytuje nám tato funkce dostatek času na podrobné pozorování a vyhodnocování výsledků přímo při běhu programu, který lze podle potřeby pozastavovat a odpadá tak případná nutnost spouštět program opakovaně kvůli sběru dalších dat a pozorování, která nebylo možné uspokojivě stihnout v reálném čase. Stejně tak je možné takto pozastavený stroj v jeho stavu uložit, virtualizační software vypnout a při příštím spuštění opětovně vše načíst do paměti a obnovit činnost virtuálního systému tam, kde byl před uložením pozastaven.

2.2.6 Izolace testovaných programů

Výhoda instalace testovaných programů na virtuálním stroji má ještě další výhodu a tou je vyšší bezpečnost, která je zajištěna izolací těchto programů od hostitelského systému na fyzickém stroji právě použitím stroje virtuálního, který takto odstíní potenciálně nebezpečný software od možnosti napáchat nějaké škody jinde než v operačním systému na virtuálním stroji, který je pro takové testování přímo vytvořen a jeho případné zotavení a návrat k původnímu stavu je záležitostí pár okamžiků a vyhýbáme se tím i případným následkům v podobě způsobených škod na osobních datových souborech, které jsou v systému uloženy.

Zmíněné odstínění je zajištěno virtualizací HW, kdy virtualizační software poskytuje operačnímu systému rozhraní k HW komponentám, které jsou ovšem všechny pouze simulované. Operační systém proto může běžet bez úprav, jako by byl nainstalován na skutečném fyzickém stroji a měl přístup k celému HW počítače, ale komunikaci se skutečným HW zprostředkovává až virtuální stroj a prostředí běhu virtualizovaného systému je odděleno od prostředí systému na fyzickém stroji, proto by nemělo dojít k ohrožení počítače, na kterém virtuální stroj běží [21]. S případným spuštěním škodlivého kódu ve virtuálním stroji dojde s největší pravděpodobností pouze k nakažení virtualizovaného OS a s takovou situací se lze velice jednoduše vypořádat návratem stroje do předchozího uloženého stavu. Je zde určitě jisté riziko v podobě možnosti výskytu chyby ve virtualizačním software, která by mohla být otevřenými vrátky do operačního systému na fyzickém PC, na kterém virtualizační software běží, ale stále je riziko menší, než v případě, že nedůvěryhodný software budeme spouštět přímo na fyzickém stroji, kde neexistuje žádný mezičlánek, který je izolační vrstvou mezi nedůvěryhodným software a operačním systémem ve skutečném počítači.

2.3 Výběr programu pro bezpečnostní analýzu

2.3.1 Struktura výběrového řízení

Výběrové řízení má dvě kola. V prvním kole provedu výběr prvních 10 nejpoužívanějších (nejstahovanějších) programů podle vytvořených žebříčků získaných ze stanovených informačních zdrojů. V kole druhém bude 10 programů vybraných v prvním kole zkušebně nainstalováno. Při instalaci budou sbírány informace o činnosti instalátoru a následně po jeho ukončení i informace o činnosti nainstalovaného programu, který bude zkušebně spuštěn. Rozhodujícím kritériem je pořadí v žebříčku oblíbenosti a k výsledkům druhého kola bude přihlédnuto tzn. že může dojít k výběru aplikace, která bude mít horší oblíbenost než první v pořadí, ale bude vykazovat zajímavé chování oproti ostatním aplikacím.

2.3.2 Určení zdrojů informací pro výběrové řízení

Pro hledání vhodného programu jsem se rozhodl využít přední české servery, které se zabývají agregováním volně dostupných programů, jejich tříděním do tématických kategorií, hodnocením kvality a následnou distribucí ke svým návštěvníkům. Rozhodujícím kritériem pro výběr serverů s tímto zaměřením bude návštěvnost těchto stránek, protože hledám oblíbenou a hojně používanou aplikaci. Podle výsledků z veřejně dostupných statistik v systému pro sledování návštěvnosti stránek `NetMonitor.cz` jsem vybral následující 3 nejnavštěvovanější servery, které měly největší počet návštěvníků a zobrazených stránek za definované časové období, kterým byl 50. týden roku 2014. Servery s nejvyšším počtem návštěv jsou www.slunecnice.cz, stahuj.centrum.cz a instaluj.cz.

2.3.3 Stanovení důležitosti informačních zdrojů

Název serveru	Návštěvy 50. týden roku 2014	Zobrazení stránek
Slunecnice.cz	169 812	1 230 943
Stahuj.Centrum.cz	213 006	1 321 598
Instaluj.cz	26 313	146 848

Tabulka 2.1: Tabulka návštěvnosti jednotlivých serverů podle serveru netnonitor.cz [1]

Na základě počtu návštěv a počtu zobrazení stránek u jednotlivých serverů stanovíme důležitost poskytnutých údajů o počtech stažení nabízených programů.

Vzhledem k řádovému rozdílu v návštěvnosti podle tabulky 2.1 budeme považovat server Instaluj.cz za méně významný a soustředíme se pouze na servery stahuj.centrum.cz a slunecnice.cz. Ze serveru stahuj.centrum.cz získáme týdenní statistiky oblíbenosti prgramů, neboť tento server vykazuje nejvyšší návštěvnost. Seznam získaných programů spolu s informacemi o počtu jejich stažení porovnáme se stejnými údaji ze serveru slunecnice.cz, kde v každé kategorii vybereme nejstahovanější programy a porovnáním absolutních počtů stažení vybereme další nejoblíbenější programy, kterými doplníme původní seznam získaný ze serveru stahuj.centrum.cz. Server slunecnice.cz totiž neposkytuje přehledný výpis programů řazených podle počtu stažení za týden napříč kategoriemi, proto je nutné sestavit pořadí složitějším způsobem při kontrole každé kategorie zvlášť.

2.3.4 Sestavení pořadí oblíbenosti programů

Následujícím krokem bude seřazení takto vybraných programů podle absolutního počtu stažení a aplikace dalších pravidel na úpravu pořadí výběru a na filtrování programů.

Programy, ke kterým jsou veřejně dostupné zdrojové kódy, budou podle stanovených pravidel vyřazeny z množiny potenciálních aplikací pro detailní analýzu.

2.3.5 Průzkum chování vybraných programů

Výsledkem výběru bude množina programů, kterou budeme podrobně zkoumat. Pro otestování vhodnosti takto vybraných programů a vybrání jednoho nejvhodnějšího pro hlubší analýzu stanovíme postup, který bude zahrnovat zkušební instalaci každého programu na testovací virtuální stroj o specifické konfiguraci tak, aby všechny instalační balíky byly instalovány do systému, který se nachází vždy ve stejném počátečním stavu, a nedošlo tak k ovlivnění jednotlivých instalací. Při instalaci bude docházet ke sběru dat o změnách v systému, které instalátor provádí, a po dokončení instalace bude následovat i zkušební spuštění aplikace, při kterém se bude zjišťovat, jak se program po spuštění chová, jaké používá knihovny a případně jaké spouští další procesy nebo služby.

2.4 Metody hledání zranitelností

Vyhledávat zranitelnosti v programu lze mnoha způsoby a závisí to na druhu aplikace i hledané zranitelnosti. Můžeme se pokoušet o studium jeho kódu a hledat podezřelá místa, ve kterých by

Pořadí	Název	Stažení za týden	Poznámka
1	Kristýna	17 370	
2	DAEMON Tools Lite	12 809	
3	AVG AntiVirus FREE	11 917	
4	VLC Media Player	11 833	Opensource - vyřazen
5	Avast Free Antivirus	11 576	
6	WinRAR	11 280	
7	CCleaner	10 916	
8	Skype	10 184	
9	Firefox	9 487	Opensource - vyřazen
10	EASEUS Partition Master	9 359	
11	Adobe Flash Player	8 423	
12	Google Chrome	7 490	
13	uTorrent	7 119	

Tabulka 2.2: Tabulka nejoblíbenějšího software v ČR [9, 10]

mohlo docházet k nějakému druhu chyb způsobujících vznik zranitelností jako je třeba přetečení bufferu. Případně lze živě monitorovat chování programu, který je již spuštěn a my sledujeme jeho reakce na zadané vstupy, které vyhodnocujeme a snažíme se identifikovat slabá místa.

2.4.1 Analýza použitých dynamických knihoven

Důležitou součástí přípravy při hledání zranitelností je analýza použitých knihoven, které program načítá při svém spuštění nebo během své práce. Obvykle každý program používá hojný počet knihoven, které mu usnadňují práci s různými programovými rozhraními např. při interakci s operačním systémem, dále mohou poskytovat nějaké standardní rutinní služby, které program potřebuje využít a bylo by zbytečné je znovu implementovat v hlavním kódu aplikace. Může se jednat o podpůrné knihovny pro kompresi a dekompresi dat, implementace různých specifikací komunikačních protokolů nebo další funkcionality společné více programům.

Pro získání seznamu knihoven budeme využívat pomocné programové nástroje. Výběru těchto nástrojů se budeme věnovat v další kapitole 2.5. Budeme hledat takové nástroje, ve kterých lze pohodlně zobrazit ke každému procesu načítané knihovny, které proces používá. Knihovny z takto získaného seznamu prověříme na přítomnost známých zranitelností, které již byly objeveny a zveřejněny. Pokud se potvrdí přítomnost zranitelné knihovny ve zkoumaném programu, bude na místě takovou zranitelnost prověřit na možnost zneužití. Přítomnost knihovny obsahující zranitelnost naznačuje, že zkoumaný software je buď zastaralý nebo nebylo věnováno dost pozornosti prověření používaného kódu od třetích stran.

Velice důležitým krokem při průzkumu použitých knihoven bude zjištění, které knihovny nevyužívají Address Space Layout Randomization (ASLR). Knihovny, které tuto technologii randomizace adresy v paměti nepodporují, razantně zvyšují riziko úspěšnosti útoku na takový software, protože pokud se útočník bude snažit podniknout útok po nalezení nějaké zranitelnosti, bude pro něj knihovna, která nepodporuje ASLR ideálním místem pro zacílení útoku, neboť vzhledem k tomu,

že adresa knihovny zůstává konstantní, lze jednoduše adresy na fragmenty užitečného kódu z této knihovny použít přímo ve škodlivém kódu [13].

Dále je vhodné zkontrolovat, zda spuštěná aplikace využívá funkci Data Execution Prevention (DEP). Tato funkce umožňuje označit v systému stránky paměti speciálním příznakem, který udává, zda se na dané stránce nachází pouze data nebo se jedná o spustitelný kód. S použitím DEP lze předcházet útokům, kdy se útočník pokouší spustit kód z předem neoznačených paměťových umístění, která mají obsahovat pouze nespustitelná data [13].

2.4.2 Testování ošetření vstupů

Základním prostředkem k ovládání programu a k interakci mezi uživatelem a programem jsou vstupy do takového programu. Vstup může být realizován např. zadáním textu v příkazové řádce nebo do formuláře, když uživatel vyplní nějaká data, případně zvolí v grafickém menu požadovanou položku. Vstupem do programu mohou být ale i data z různých periferních zařízení nebo jiných programů či síťových zdrojů, pokud program podporuje síťovou komunikaci a například aktivně naslouchá příchozímu spojení na určitém portu nebo se sám pokouší spojení navázat. Příkladem vstupu do programu mohou být „roury“ (pipes) nebo „vzdálené volání procedur“ (RPC).

Vstupy do programu musí být vždy podrobeny pečlivému zabezpečení proti neočekávaným hodnotám, neboť se jedná o data, u kterých se nelze spolehnout na žádná pravidla nebo normy týkající se jejich formátu a v podstatě mohou být libovolná. V případě, že dojde k situaci, kdy program striktně nekontroluje vstupní data, může dojít k nestandardní situaci, kdy program obdrží data, na která nebude schopen adekvátně reagovat a v tuto chvíli může nastat několik možností. Program provede stanovenou operaci a v závislosti na chybných datech podá chybné, neočekávané nebo nesmyslné výsledky.

Horší situace nastává, pokud dojde při práci s nezkontrolovanými daty ze vstupu k pádu programu. V prvé řadě se jedná o značné narušení uživatelského pohodlí, protože uživatel, který mohl zadat neplatný vstup omylem bez úmyslu na program zaútočit, právě přišel o své výsledky práce, které ještě nebyly uloženy v souboru na pevném disku. Z hlediska bezpečnosti se jedná o potenciální zranitelnost v programu, která může být zneužita například ke spuštění kódu zadaného útočníkem.

2.4.3 Statická analýza kódu aplikace

Statickou analýzou kódu rozumíme analýzu kódu bez jeho spuštění za účelem zjištění, co zkoumaný kód provádí za operace a z toho aspoň částečně odvodit jeho funkci. Při tomto druhu analýzy obvykle používáme nástroje nazývající se disassembler, které umí přeložit binární kód aplikace do lidsky čitelné podoby jazyku symbolických instrukcí. Takto reprezentovaný program je již mnohem čitelnější a se znalostí instrukcí dané platformy, pro kterou byl program přeložen, je možné začít s analýzou.

Proti analyzování programu se někdy vývojáři a vydavatelé softwaru snaží bránit, a proto používají nástroje, které programový kód zašifrují, komprimují či obfuskují. Takové programy je pak složitější staticky analyzovat, protože skutečný kód aplikace, který nese funkcionalitu celého programu, začne existovat až po jeho rozbalení a po spuštění, kdy dojde teprve k rozšifrování do paměti počítače. Kód nalezený při statické analýze je kódem programu, který má za úkol při spuštění provést dešifrování resp. dekomprimaci původní aplikace. Program, který byl rozbalen, lze následně získat z paměti počítače a pak provést statickou analýzu rozbaleného kódu, ale tento proces je už

složitější než prosté disassemblování původního binárního souboru s využitím vhodného nástroje a může méně zkušeným vývojářům znepříjemnit případně i znemožnit proces analyzování kódu aplikace. Použití některých nástrojů pro komprimaci lze rozpoznat podle vzorků kódu obsažených ve výsledném binárním souboru, který v sobě zahrnuje rozbalovací podprogram vytvářející při spuštění kód hlavního programu. Zabalením programu se může zmenšit jeho výsledná velikost, a to je výhodou zejména v případech distribuce škodlivého kódu, kdy je nutné program přenést nepozorovaně a rychle do cílového stroje. Pro zjištění, zda-li je zdrojový program zabalen, slouží nástroj zmíněný v sekci Použitý software 2.5 [12].

2.4.4 Dynamická analýza

Při analýze programu za běhu opět disassemblujeme kód, aby byl lépe čitelný. Získáme kód v jazyce symbolických instrukcí, ale na rozdíl od statické analýzy, kde kód pouze čteme a snažíme se rekonstruovat původní zdroj, abychom odhalili celkovou funkci programu, spustíme zkoumaný program v ladícím nástroji a můžeme tak přímo pozorovat vnitřní stavy programu, sledovat jeho běh, jak se vyhodnocuje tok programu a skoky a jaká data obsahuje alokovaná paměť. Tyto dodatečné poznatky, které analýzou za běhu získáme, nám poskytují cenné informace o programovém kódu, abychom mohli lépe odhadnout jeho celkovou funkci v kontextu zkoumaného software [12].

Při takové analýze můžeme programu předkládat různé vstupy a následně pozorovat jeho chování nejen z pozice uživatele, který vidí pouze zpětnou vazbu, kterou mu samotný program poskytne v rámci svého rozhraní, ale můžeme pozorovat program z pozice samotného stroje, který vykonává jeho instrukce. Při testování můžeme i zasahovat do chodu programu a provádět změny hodnot v registrech nebo v paměti mezi vykonáváním jednotlivých instrukcí a tím otestovat odolnost programu podle toho, jak se chová při výskytu neočekávaných hodnot a různých chyb, které vlivem těchto změn z venčí mohou být vyvolány.

2.4.5 Síťová komunikace

Síť umožňuje programům spojení s dalšími zařízeními. Dnes je vzhledem k rozšíření dostupnosti internetového připojení komunikace programů po síti rozšířenou záležitostí. Komunikace po síti může být klíčovou funkcí programu, ale může sloužit i pro servisní činnost jako jsou instalace programových součástí, doplňků nebo případné aktualizace na novější verzi.

V závislosti na povaze přenášených dat je zapotřebí zajistit určitý stupeň zabezpečení. V případě, že se jedná o veřejná data, u kterých není riziko jejich odcizení nebo neoprávněné změny při přenosu k uživateli, není potřeba klást na zabezpečení veliký důraz, ovšem v případě, že jsou po síti přenášeny citlivé údaje, je na místě jejich odpovídající zabezpečení proti odcizení nebo změnám, aby bylo zabráněno případným škodám spojených s únikem nebo poškozením přenášených informací mezi zúčastněnými stranami.

Při analýze komunikace potřebujeme zachytávat data proudící po síti, která následně můžeme podrobit zkoumání z hlediska jejich struktury, obsahu a zabezpečení. Pro monitorování komunikace existuje nástroj s názvem Wireshark, který jsem popisoval v předchozí kapitole o výběru software pro účely této práce. S pomocí tohoto programu je možné zachytávat veškerou komunikaci, která proudí mezi vnější sítí a kontrolovaným strojem, a poskytovat přehledné záznamy o odeslaných a přijatých packetech. Z odposlechnutých dat je pak možné získávat poznatky o druhu a účelu síťové komunikace zkoumaného programu a jejich podrobnou analýzou objevit případná slabá místa,

která by mohla být potenciálním zdrojem bezpečnostních hrozeb, které mohou poskytnout útočníkovi příležitost k napadení cílového systému. Z pohledu bezpečnosti je nejdůležitější identifikovat přenosy citlivých dat a ověřit případně úroveň zabezpečení a zhodnotit, zda je tato úroveň dostatečně vysoká vzhledem k povaze přenášených informací. Pokud se ukáže, že v této oblasti existuje slabé místo, následuje prověření takového místa na odolnost proti předpokládaným typům možného útoku.

2.5 Použitý software

Pro analýzu běhu programu, jeho případné síťové komunikace, zjištění podrobností o načítaných spustitelných modulech a dalších potřebných informací bylo potřeba vybrat vhodné programové nástroje, které by umožnily potřebná data získat a poskytnout pro další zpracování. Při výběru těchto nástrojů jsem čerpal zejména z knihy [12] následně pak z dalších online zdrojů vztahujících se přímo k uvedeným programům nebo obecně k nástrojům pro reverzní analýzu programů.

2.5.1 Microsoft Sysinternals Suite

Microsoft Sysinternals Suite je soubor programových nástrojů pro pokročilé monitorování činnosti operačního systému Windows a procesů v něm operujících, které vytvořili Bryce Cogswell a Mark Russinovich [19]. Dnes tyto nástroje poskytuje přímo Microsoft na svých stránkách[1], a to i ke spuštění online bez nutnosti je stahovat.

2.5.1.1 Process Explorer

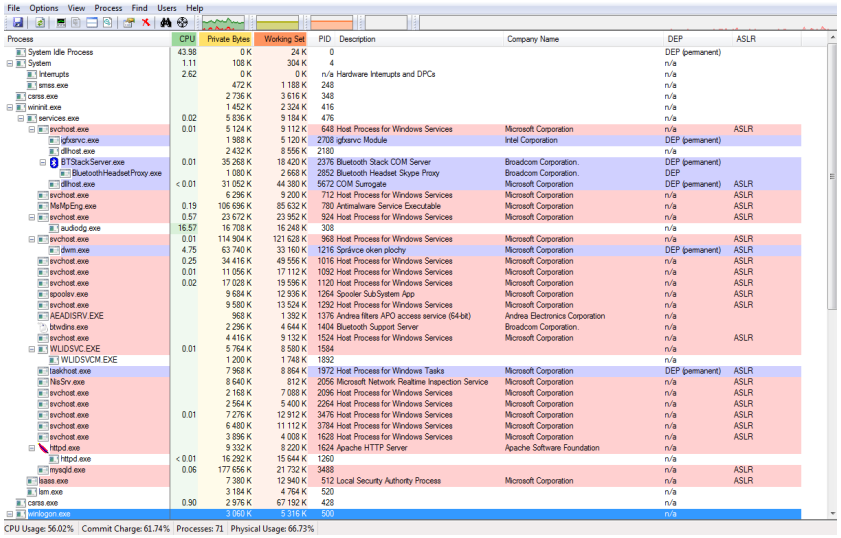
`Process Explorer` je prvním důležitým nástrojem, který bude při této práci hrát důležitou roli. Jedná se o nástroj pro monitorování spuštěných procesů v reálném čase. Funkce je podobná vestavěnému nástroji `Správce úloh`, ale možnosti nástroje Process Explorer jsou mnohem rozsáhlejší. V horní části okna programu se nachází grafy využití CPU, paměti, aktivity vstupů a výstupů a zatížení GPU. Process explorer ve svém okně zobrazuje přehledný výpis běžících procesů a služeb ve stromové struktuře, takže je na první pohled vidět, který proces byl spuštěn jiným procesem. V dalších sloupcích je pak vidět vytížení CPU, množství alokované paměti, identifikátor procesu a jeho popis. Lze zobrazit i další sloupce jako údaje o použití DEP[2] nebo ASLR[3] apod. Každý řádek navíc může být v pozadí zabarven různými barvami jak je vidět na obrázku 2.3, které poskytují další detailní informaci o procesu samotném. Zabarvení řádků je nastavitelné [8].

V hlavním okně v jeho spodní části ve stavovém řádku se pak nachází další užitečné číselné údaje jako procentuální vyřížení CPU a paměti a celkový počet běžících procesů. Náhled na programové okno Process Exploreru je vidět na obrázku 2.3.

[1] https://technet.microsoft.com/en-US/sysinternals

[2] Data Execution Prevention je technologie, která zabraňuje spuštění dat v paměti, která není předem označena jako paměť obsahující spustitelný kód.[13]

[3] Address space layout randomization provádí náhodné rozmístění zdrojových kódu spouštěných aplikací, zásobníku, haldy i načítaných spustitelných modulů do paměti počítače. Cílem je znemožnit nebo aspoň razantně ztížit útoky typu buffer overflow, které by využívaly znalost adresy v paměti určité knihovny nebo části kódu.[13]



Obrázek 2.3: Okno programu Process Explorer ze Sysinternals

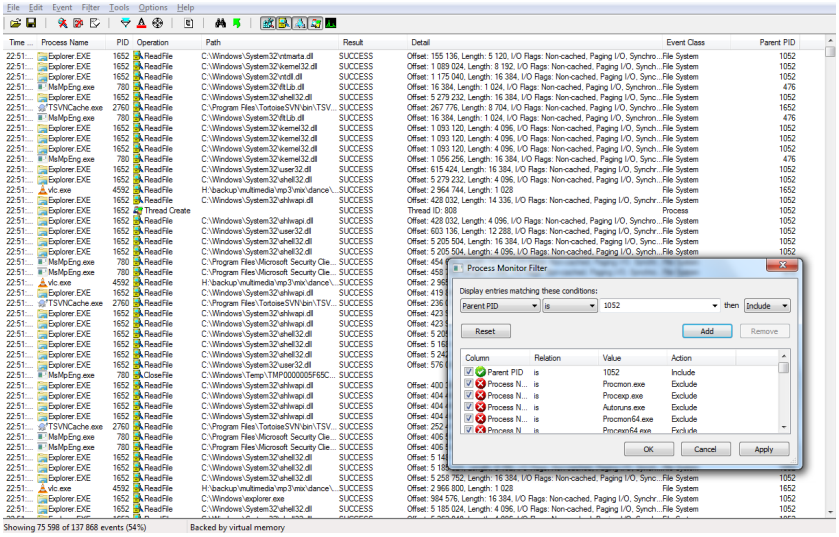
Mezi pokročilé funkce skryté pod položkami v hlavním menu patří mimo jiné i velice užitečná možnost zobrazit všechny načtené spustitelné moduly pro každý proces spolu s množstvím informací o modulech samotných včetně jejich fyzického umístění v souborovém systému a dostupných podrobností. Je tak možné zjistit, které načtené knihovny používají například ASLR nebo které jsou digitálně podepsané vydavatelem.

Další funkcí je zobrazení aktivních ukazatelů na otevřené soubory, adresáře, události nebo sezení, kdy je možné přehledně zobrazit k jakým zdrojům aplikace přistupuje. Process Explorer umožňuje v těchto položkách i vyhledávat a to napříč spuštěnými procesy. V případě, že potřebujeme najít, které procesy používají daný zdroj, stačí ho pomocí metody `find handle or dll` vyhledat. A v případě, že nějaký proces má například aktivní přístup k adresáři, který nebyl korektně uzavřen a kvůli tomu tento adresář nelze odstranit nebo přesunout, existuje zde příkaz pro uzavření takového handle manuálně z Process Exploreru, kterým se problém vyřeší.

2.5.1.2 Process Monitor

Dalším užitečným programem z balíku Sysiternals je Process Monitor, který také monitoruje činnost systému a v něm běžících procesů, ale je určen narozdíl od Process Exploreru zejména pro následnou analýzu záznamu, který je během monitorování vytvořen a uložen do souboru. Process Monitor totiž zaznamenává všechny systémové události, kam patří veškerá interakce spuštěných procesů s operačním systémem. V záznamu jsou vidět přístupy do registru, manipulace s jednotlivými klíči, přístupy k souborům a vytváření a rušení nových procesů nebo vláken. Každý řádek obsahuje jednu událost zachycenou při běhu Process Monitoru.

Ve velkém počtu událostí by bylo složité se vyznat, proto Process Monitor disponuje funkcí pro pokročilé filtrování událostí, které je vidět na obrázku 2.4. Ve filtru lze definovat sadu podmínek, které se pak aplikují na nasbíranou kolekci dat. V základním nastavení jsou definované podmínky, které z výpisu rovnou vyřazují nezajímavé události, které například vytváří sám diagnostický nástroj



Obrázek 2.4: Okno programu Process Monitor ze Sysinternals

Process Monitor. Další uživatelem definované podmínky lze vytvářet velice jednoduše a lze je aplikovat na všechny dotupné sloupce včetně těch, které nejsou aktuálně vypsané v přehledu. Při volbě operace jsou zde možnosti porovnání hodnoty včetně nerovností ale i operací pro filtrování podle řetězců, které umožňují vyhledávat záznamy obsahující v některém ze sloupců hledaný řetězec nebo pouze jeho část. Lze vytvořit i opačnou podmínku. Celá operace ve filtru má pak strukturu sloupec s daty — operace — hodnota pro porovnání — akce. Akce má pouze dvě volby, které udávají zda se řádek splňující definovanou podmínku má v přehledu zobrazit, nebo naopak skrýt. S pomocí tohoto pokročilého filtru je možné provádět speciální a složitě definované výběry událostí i v tak velkém množství dat jaké obsahují záznamy z interakce procesů s operačním systémem.

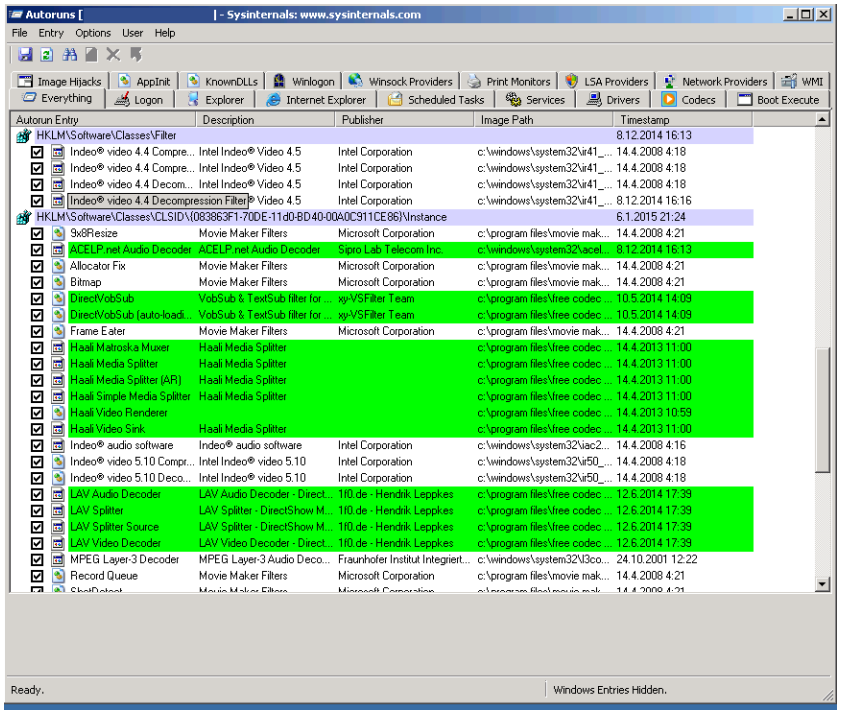
Ke každé události je k dispozici množství detailních informací, které lze zobrazit přehledně roztříděné na kartách s vlastnostmi události dostupných z kontextového menu. Výsledný seznam událostí lze exportovat do souboru, a to celý nebo jen položky, které odpovídají nastavení filtru. Kdykoliv později je tak možné se vrátit k analýze pořízeného záznamu.

2.5.1.3 Autoruns

Nástroj Autoruns je užitečným pomocníkem při průzkumu systémových registrů za účelem vyhledání záznamů souvisejících s automatickým spouštěním úloh po startu systému. Jeho uživatelské rozhraní je vidět na obrázku 2.5. Umí třídit záznamy do kategorií, které přehledně zobrazuje na jednotlivých záložkách a dovede také uložit výpis nalezených záznamů do souboru. Uložené soubory lze v programu porovnávat mezi sebou a hledat efektivně rozdíly v jejich obsahu. To umožňuje nalézt nové záznamy v registru, které byly vytvořeny v době mezi uložením porovnávaných výpisů.

2.5.2 Dependency Walker

Tento nástroj umožňuje zjistit, jaké závislosti analyzovaný program obsahuje. Jedná se o velice užitečnou pomůcku při průzkumu používaných dynamicky načítaných knihoven. Narozdíl od nástroje

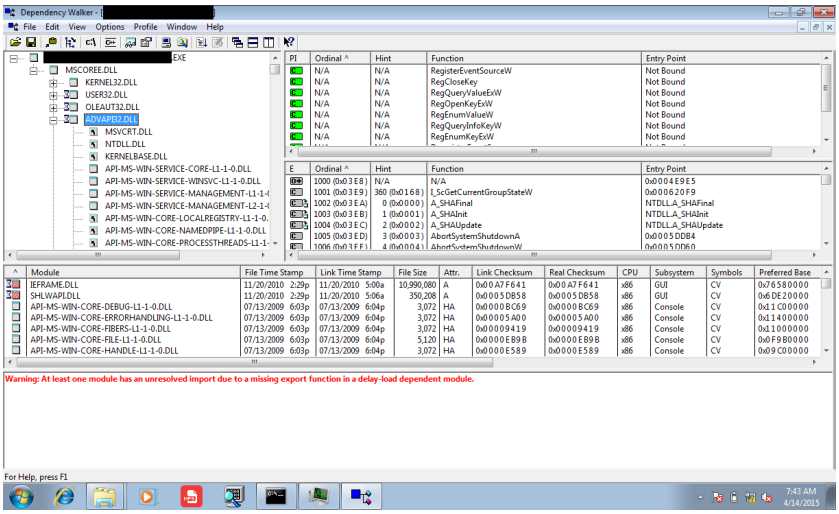


Obrázek 2.5: Okno programu Autoruns ze Sysinternals

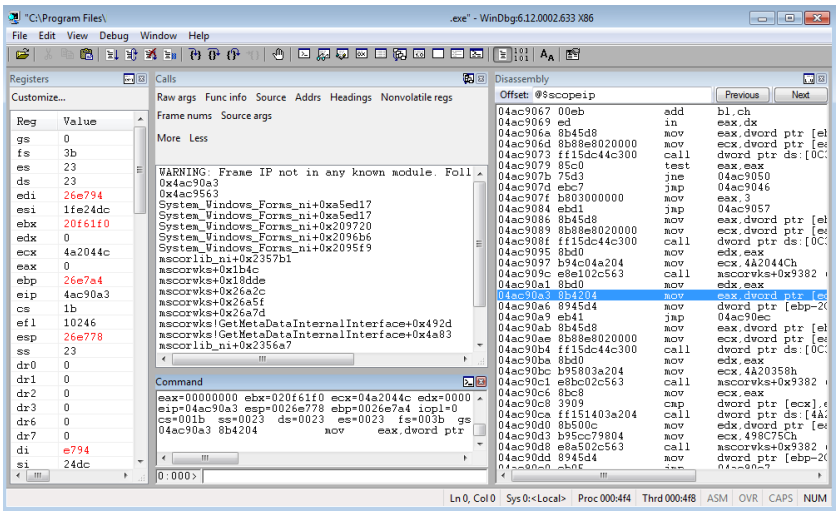
Process Explorer, který je schopen zobrazit aktuálně spuštěné a načtené moduly, které aplikace využívá, provede tento software kompletní stromový výpis všech nalezených závislostí rekurzivně do více úrovní. Je tak velice jednoduše možné najít případné problémy v těchto závislostech, které mohou být způsobeny chybějícími knihovnami v systému, nedokončenou nebo jinak poškozenou instalací software a ovladačů, chybným nastavením cest ke sdíleným knihovnám a dalšími příčinami, které způsobí, že software nepracuje správně nebo vůbec nelze spustit. Pro účely této práce bude nejvýznamnější funkcí získání podrobného seznamu používaných knihoven.

2.5.3 WinDBG

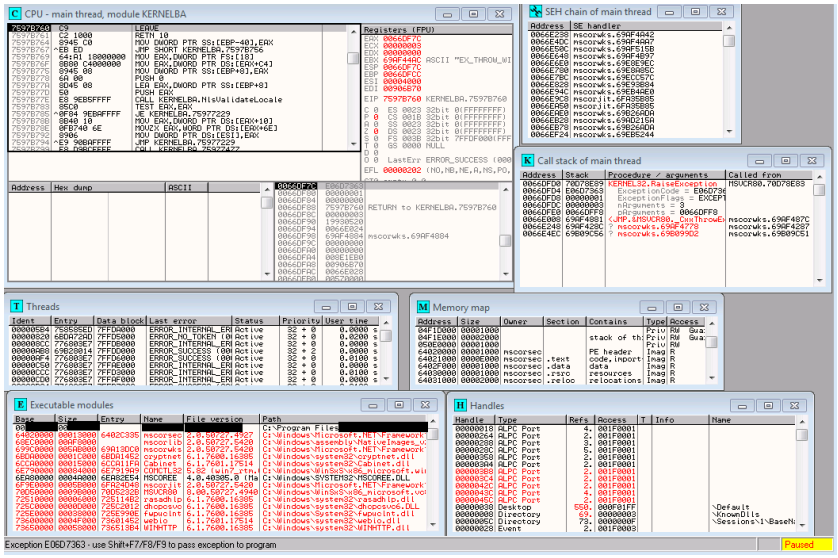
WinDBG, který je součástí balíku `Debugging Tools for Windows`, je programem pro disassemblování kódu aplikace a jeho ladění za běhu. Program disponuje mnoha funkcemi, které umožňují zobrazovat i upravovat hodnoty aktuálně uložené v registrech, procházet disassemblovaný zdrojový kód, sledovat zanoření volání podprogramů a příslušných rámců na zásobníku, kontrolovat spuštěná vlákna, analyzovat obsah alokované paměti, monitorovat načítané dynamické knihovny a další spustitelné moduly v rámci aplikace a celkově sledovat a řídit běh zkoumaného laděného programu. Při krokování programu jsou případné změny hodnot v registrech mezi jednotlivými instrukcemi zvýrazněny červenou barvou.



Obrázek 2.6: Okno programu Dependency Walker



Obrázek 2.7: Okno programu WinDBG



Obrázek 2.8: Okno programu OllyDBG

2.5.4 OllyDBG

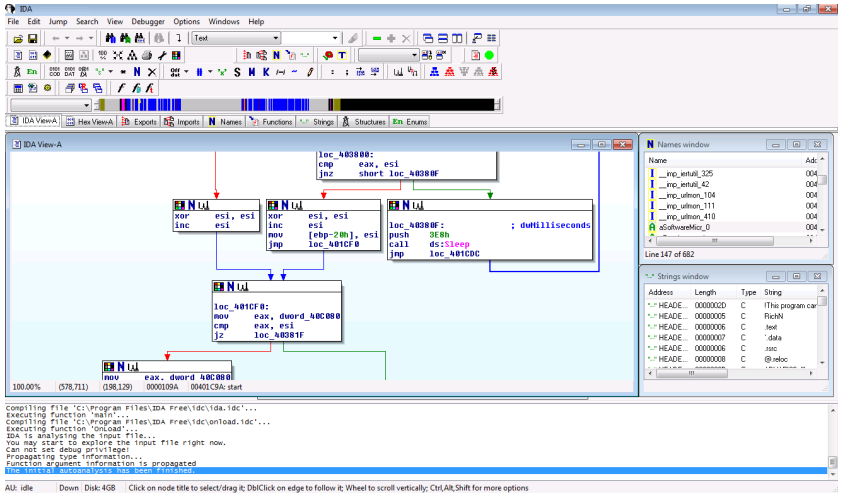
Podobným nástrojem jako je WinDBG je nástroj OllyDBG, který rovněž poskytuje možnost ladit program za běhu a disassemblovat jeho kód do lidsky čitelné podoby jazyku symbolických instrukcí. Disponuje funkcemi pro řízení běhu programu jako je krokování, nastavování breakpointů, zobrazování důležitých dat poskytující informace o vnitřním stavu programu jako je výpis registrů se zvýrazňováním změn v jejich hodnotách, výpis paměti, zobrazení spuštěných vláken, hierarchii rámců zásobníku a načtených modulů v laděné aplikaci [12].

Ovládání programu se provádí zejména přes grafické uživatelské rozhraní, nabídky hlavního menu a kontextovou nabídku v rámci pracovního okna programu, které je vidět na přiloženém obrázku 2.8.

2.5.5 IDA Pro 5.0

Interactive Disassembler IDA je značně pokročilý software pro ladění a zkoumání aplikací, který obsahuje podporu pro množství architektur procesorů i pro mnoho používaných formátů spustitelných souborů a může být provozován napříč platformami (Linux, Windows, OS X). Disponuje množstvím funkcí pro analýzu kódu, které dovedou sestavit diagramy poskytující informace o provázanosti jednotlivých metod a jejich programového rozhraní. Velikou výhodou je, že tyto diagramy jsou vysoce interaktivní a dovolují tím pohodlně, rychle a efektivně provádět rozbor kódu, při kterém obsluze napomáhá chytré zvýrazňování souvisejících zobrazovaných informací, překlady a přepočty adres s náhledy do příslušných oblastí paměti a další užitečné a velice chytře propracované funkce [12]. Tento program je poskytován v omezené verzi na vyzkoušení zdarma, ale jeho plná profesionální verze je dostupná od 1129 USD[4].

[4]Informace na stránce https://www.hex-rays.com/products/ida/order.shtml získaná dne 6. 4. 2015



Obrázek 2.9: Okno programu IDA Pro

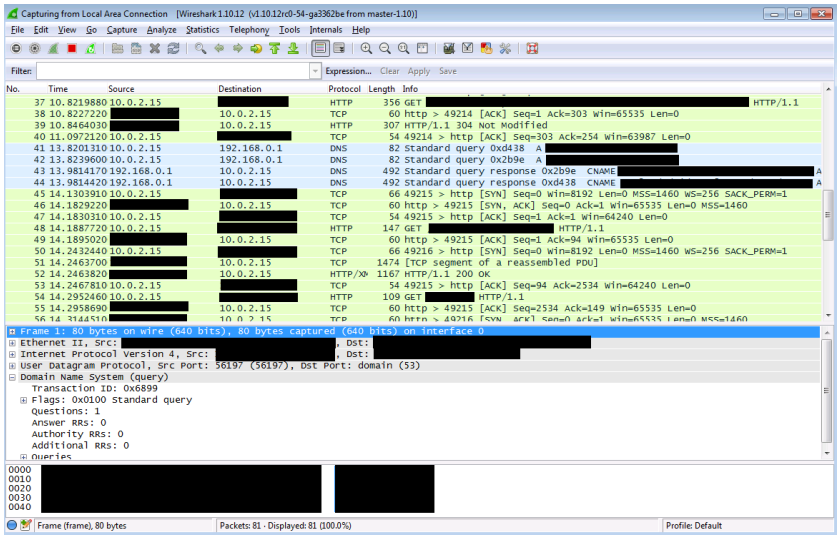
2.5.6 Wireshark

Wireshark je mocným nástrojem pro sledování síťové komunikace, která se pohybuje přes určené síťové rozhraní v počítači. Lze s ním monitorovat všechny packety, které jsou odesílány a přijímány aplikacemi spuštěnými na hostitelském počítači. Pokud to síťové rozhraní umožňuje, umí jej Wireshark přepnout do tzv. promiskuitního módu, a tím zachytávat veškerou komunikaci proudící přes toto zařízení včetně komunikace, která není přímo určena pro rozhraní s danou adresou. Tento software dovede komunikaci zachytávat, zaznamenávat a poskytovat přehledně připravené výstupy, ve kterých je u každého zaznamenaného packetu uvedeno množství užitečných informací, jako je protokol, zdrojová a cílová IP adresa a v detailním rozboru se nachází podrobné informace o všech protokolech v jednotlivých síťových vrstvách. Packety jsou i logicky provázány a Wireshark umí seskupovat fragmentovaná data do výsledné podoby, která byla přijata a dovede poskytnout informace o vazbách mezi odesílanými požadavky a doručenou odpovědí od protistrany. Zároveň má vestavěné funkce pro filtrování a řazení nasbíraných dat [11].

2.6 Stanovení postupu testování programů

Na základě předchozí analýzy, ve které jsme stanovili kritéria pro výběr testované aplikace, vhodného testovací prostředí, metod pro bezpečnostní analýzu aplikace a potřebných nástrojů, které nám při analýze poskytnou důležité údaje o chování aplikace, sestavíme postup pro testování jednotlivých programů tak, abychom byli schopni zjistit potřebné vlastnosti testovaných programů a na jejich základě zhodnotit splnění stanovených kritérií.

U všech programů bude zaznamenán průběh instalace a pozorování spolu se sesbíranými daty o činnosti spuštěných procesů a vyvolaných událostech z monitorovacího programu Process Monitor a s daty z programu Autoruns, který poskytuje seznam položek v registrech v oblasti pro automatické spouštění úloh po startu systému. Takto získaný seznam je možné s využitím funkce umožňující porovnání dvou seznamů porovnat s výpisem stejné oblasti systémových registrů, který bude vytvořen



Obrázek 2.10: Okno programu Wireshark

ve výchozím stavu systému a uložen do souboru, abychom mohli porovnat rozdíly mezi těmito dvěma stavy před instalací a po instalaci testovaného programu.

Při monitorování instalátoru budeme sledovat zejména zápisy do systémového registru operačního systému (události obsahující ve svém názvu RegCreate, RegSet), instalaci dalších součástí, které mohou pro uživatele představovat potenciálně nechtěné aplikace[5] a pokusy o přístup na internet a navigaci uživatele na webové stránky vydavatele za účelem zobrazení reklamy nebo nabídky dalšího software a získání údajů o uživateli.

Po sesbírání dat z testovacích instalací bude následovat jejich vyhodnocení a na základě stanovených kritérií bude vybrán jeden program, který bude podroben podrobnější bezpečnostní analýze.

2.6.1 Sestavený postup testování

- 1. Spustit systém obnovený do počátečního stavu ze snímku (`snapshot` ve Virtual Box) potřebné nástroje jsou v tomto stavu již spuštěny (Process Explorer, Process Monitor).
- 2. spustit záznam v process monitoru a sledovat Process Explorer;
- 3. spustit instalaci programu;
- 4. v procesu instalace ponechat vše ve výchozí konfiguraci;
- 5. po ukončení instalace uložit záznam z process monitoru;

[5]PUP / PUA — Potentially unwanted program / application je software, který uživatel nepotřebuje, ale jeho stažení a instalaci provedl spolu s jiným programem často proto, že si nepřečetl licenční ujednání, ponechal možnosti instalace ve výchozím či doporučeném nastavení, které obsahuje označenou žádost o získání takového software. Tento software je pro uživatele obvykle úplně zbytečný, pouze zabírá systémové prostředky a je zdrojem zranitelností případně rovnou může obsahovat škodlivý kód, který o uživateli získává osobní informace, ovlivňuje výsledky hledání na webu, zobrazuje nevyžádanou reklamu a provádí další operace, které uživatele obtěžují nebo poškozují [17].

- 6. restart systému;
- 7. spustit nástroj Autoruns a zjistit změny v registrech v oblasti pro automatické spuštění úloh po startu systému;
- 8. zkontrolovat v aplikaci Process Explorer běžící procesy.

Po vyhodnocení výsledků získáme potřebná data, na základě kterých určíme jednu vybranou aplikaci ze všech testovaných. Tato aplikace bude předmětem bezpečnostní analýzy, ve které se budeme podrobně zabývat celým procesem instalace aplikace, následně její stabilitou při provozu a hlavně prověřením zabezpečení síťové komunikace.

Kapitola 3

Realizace

V této kapitole se budeme věnovat přípravě prostředí pro instalaci, spouštění a testování zkoumaných aplikací. Následně se budeme zabývat samotným testováním chování aplikací na základě vytvořeného postupu v rámci předchozí analýzy, abychom po dokončení stanovených kroků postupu mohli vybrat jednu aplikaci ze všech testovaných a provést její bezpečnostní analýzu. Bezpečnostní analýza bude zahrnovat prověření stability programu a úrovně zabezpečení síťové komunikace při instalaci i při provozu nebo případné aktualizaci za účelem nalezení možných zranitelností, které by mohly být zneužity pro útok.

3.1 Virtuální prostředí

Pro testování jednotlivých programů budeme používat virtuální počítač s definovanými parametry, které vyplynuly z analýzy.

Vybraný virtualizační software Virtual Box je k dispozici ke stažení z oficiálních stránek na adrese `www.virtualbox.org`. Software je publikován pod licencí GNU General Public License V2 a je proto volně dostupný včetně svých zdrojových kódů. Virtual Box je k dispozici pro 4 platformy a to OS Windows, OS X, Linux a Solaris. Pro Windows a OS X jsou k dispozici i verze pro 32bitové nebo 64bitové verze OS.

Fyzický stroj, na kterém probíhalo testování, disponoval procesorem Intel Core2Duo 1.4 GHz a operační pamětí o velikosti 4 GiB. Virtuálnímu stroji jsem po předchozích zkušenostech přidělil virtuální pevný disk o fixní velikosti, aby nedocházelo ke zbytečné fragmentaci souboru virtuálního disku a možnému poklesu výkonu při nárůstu jeho velikosti. Z dostupných systémových prostředků dostal virtuální stroj s 32bitovým OS Windows 7 pro svůj běh k využití jedno celé jádro fyzického CPU bez procentuálního omezení dovoleného zatížení, které bylo nastaveno na 100%, a 1 GiB operační paměti RAM.

Pro účely testování jsme vytvořili virtuální pevný disk, na který jsme nainstalovali operační systém Windows 7 Professional ve 32bitové verzi. Následně jsme operační systém doplnili o potřebný diagnostický software a upravili jsme některá nastavení, která neměla vliv na testované programy, ale například na výkon systému. Proto byly vypnuty pokročilé grafické efekty a podobné nepotřebné záležitosti, které mají význam spíše estetický a svým vlivem se nepodílí přímo na funkci systému tak, aby ovlivnily výsledky této práce. Takto vytvořený pevný disk byl zálohován jako zdroj pro

vytvoření duplikátu virtuálního stroje, aby nebylo v případě výskytu nějaké fatální chyby při experimentech s nastavením OS i virtuálního stroje nutné celou instalaci provádět znovu od začátku.

Pro zjednodušení výměny bylo zřízeno v hostitelském systému pro virtuální stroj sdílené úložiště, které bylo připojeno jako virtuální jednotka, aby bylo možné pohodlně přenášet potřebné soubory, zejména objemnější instalační balíky nebo záznamy a protokoly ze sledování práce systému při testování, mezi virtuálním a hostitelským systémem. Rovněž byla povolena sdílená schránka pro možnost přenosu menších textových dat např. příkazů nebo hodnot z diagnostických programů, běžících na virtuálním systému. Pro zpřístupnění těchto pokročilých funkcí bylo nutné do virtuálního systému doinstalovat balíček Přídavky pro hosta (angl. Guest Additions), který umožňuje jednoduše aktivovat výše zmíněné užitečné funkce. Jeho instalace je jednoduchá a provádí se po rozběhnutí virtuálního systému z hlavního menu okna virtuálního stroje, kde je možné připojit do mechaniky virtuálního OS speciální obraz disku s tímto balíčkem.

V počátečním stavu systému byl vytvořen snímek ve virtualizačním programu Virtual Box, který byl výchozím stavem pro každou instalaci testovaného programu. Tímto byl zajištěn požadavek na stejný výchozí stav systému pro každý test a samotný návrat systému do tohoto počátečního stavu trval obvykle v řádu desítek sekund.

3.2 Druhé kolo výběrového řízení

Na připraveném virtuálním stroji jsem postupně prováděl podle stanovených postupů v analýze této práce instalaci každého jednotlivého programu vybraného v prvním kole a podle instrukcí jsem prováděl záznam údajů potřebných pro následné vyhodnocení jednoho vítězného kandidáta na podrobnou bezpečnostní analýzu. Záznamy v registru označují počet operací, které souvisely se zápisem do systémovéo registru. Měření počtu záznamů do registrů bylo provedeno nástrojem Process Monitor z balíku Microsoft Sysinternals Suite. Sloupec Autoruns označuje počet záznamů, které přibyly ve výpisu úloh naplánovaných po startu systému po instalaci programu a které byly zjištěny nástrojem Autoruns rovněž z balíku Microsoft Sysinternals Suite. Sloupec PUP označuje, zda byly instalovány nebo nabízeny potenciálně nechtěné aplikace. Tato informace byla získána na základě pozorování instalačního procesu uživatelem a v případě, že instlační program zobrazil nabídku k instalaci dalších přídavných programů, má tento parametr v tabulce 3.2.1 hodnotu „Ano“ v ostatních případech je hodnota „Ne“. Sloupec Web udává, jestli instalátor směroval uživatele na nějakou webovou stránku. Tento údaj byl opět zjištěn pozorováním instalačního procesu uživatelem a pokud došlo k automatickému otevření okna prohlížeče během instalace nebo po jejím ukončení, má tento parametr hodnotu „Ano“. V ostatních případech je hodnota „Ne“.

3.2.1 Naměřené výsledky

Název programu	Záznamy v registru	Autoruns	PUP	Web
Kristýna	34 654	21	Ano	Ano
DAEMON Tools Lite	856	5	Ne	Ano
AVG AntiVirus FREE	5 269	21	Ne	Ne
Avast Free Antivirus	33 723	29	Ne[a]	Ne
WinRAR	583	3	Ne[b]	Ne
CCleaner	14 809	16	Ne[a]	Ne
Skype	5 531	4	Ne	Ne
EASEUS Partition Master	158	6	Ne	Ano
Adobe Flash Player	14 520	18	Ne[a]	Ano
Google Chrome	12 973	15	Ne	Ne

Tabulka 3.1: Tabulka výsledků testování chování programů

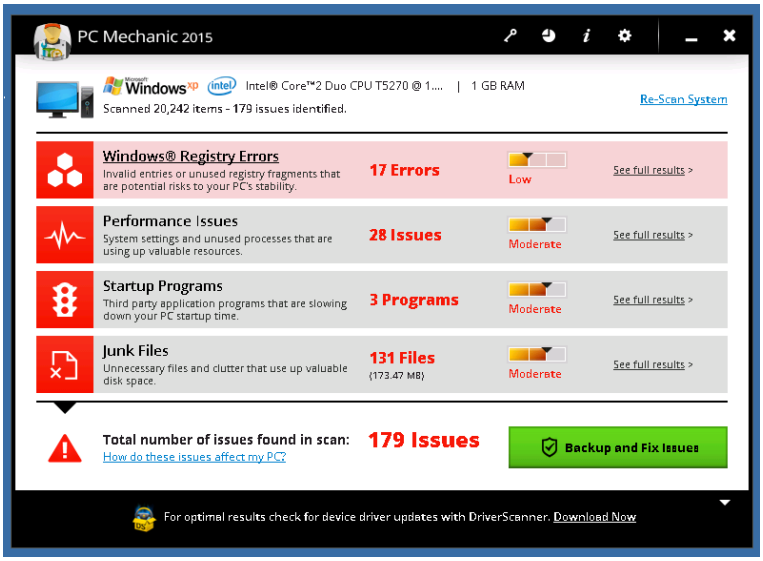
[a] Google Chrome je samostatně nabízená a známá aplikace pro prohlížení webu, která není označena za PUP, Google Toolbar už tak jednoznačně užitečný není.

[b] Seznam lištička je známý toolbar od společnosti Seznam.cz, a.s. a i při odsouhlasení instalace se pouze instalátor tohoto přídavku rozbalil do složky C:\Users\<uživatelské jméno>\AppData\Local\Temp, nainstalovat by ho bylo potřeba ručně.

3.2.2 Hodnocení výsledků

Změny provedené v systémových registrech v oblasti záznamů pro automatické spouštění úloh po startu operačního systému byly různé povahy. V případech programů jako AVG AntiVirus FREE, Avast Free Antivirus a Skype, které jsou vydávány renomovanými společnostmi, se záznamy zjištěné nástrojem autoruns týkaly výhradně úloh spojených s činností instalovaných programů. V případě dalších zkoumaných programů se v jisté míře objevovaly záznamy naznačující spouštění různých skrytých procesů kontrolujících aktualizace nebo provádějící blíže neznámou činnost.

Program Kristýna byl podle výsledků nejoblíbenějším mezi uživateli a zároveň se projevil jako nejzajímavější z hlediska svého chování, protože zásah do registru operačního systému byl rozsáhlý a bylo přidáno velké množství nových klíčů a v průběhu instalace nabízel uživateli PUP a dokonce při opakovaných pokusech o instalaci tohoto programu se nabídka přídavných programů měnila. Mezi nabízenými programy byl i software jako webový prohlížeč Opera nebo komunikační nástroj Skype, které jsou známé a považují se za bezpečné. Ovšem mnohem zajímavějším produktem nabízeným k instalaci byl program s názvem PC Mechanic, který je ukázkovým případem PUP a který opravdu překvapil, když se ihned po dokončení instalace spustil a začal provádět kontrolu počítače a v čisté instalaci OS Windows 7 se mu podařilo nalézt poměrně velké množství hrozivě vypadajících chyb. Vyhledal jsem na internetu reference na tento podezřelý software a zdá se, že je příkladem PUP, který se snaží zapůsobit na nezkušené uživatele a přimět je k nákupu produktu, který nepotřebují a dokonce může uživatelům, kteří budou program používat, způsobit problémy a poškodit operační systém nesprávnými zásahy do systémových registrů a dalších oblastí tak, že celá operace bude kontraproduktivní a místo slibovaného zrychlení a vyčištění systému dojde k jeho zpomalení nebo dokonce k jeho havárii vlivem těchto zásahů.



Obrázek 3.1: PC Mechanic 2015

Společnost Microsoft, Corp. se k programům tohoto druhu, které slibují vyčištění a opravu systémových registrů, obecně vyjádřil v jednom ze článků na webových stránkách podpory, kde přímo uvádí, že „společnost Microsoft nepodporuje používání nástroje pro čištění registru. Některé programy volně dostupné na internetu mohou obsahovat spyware, adware nebo viry“.[15] PC Mechanic 2015 resp. vydavatel tohoto software `http://www.uniblue.com/` uvádí na svých webových stránkách, že je Microsoft Partner (Gold application development), ovšem po průzkumu webu answers.microsoft.com na klíčové slovo uniblue jsem nalezl několik diskuzních vláken[4, 5], kde se uvádí, že produkty od tohoto vydavatele nejsou bezpečné a jedná se o neužitečný software, který může napáchat více škody než užitku a být pro uživatele zdrojem problémů.

Proto není používání takových programů doporučeno. V tomto případě čistícího nástroje s názvem PC Mechanic 2015, který našel tolik závažných problémů hned po instalaci operačního systému, nebudí takové hlášení mnoho důvěry, protože je málo pravděpodobné, že by nová instalace systému obsahovala tolik chyb, které program hlásí a vydává je za závažné nedostatky, které potřebují okamžitě opravit. Když k tomu uvážíme, že se jedná o obchodní model postavený na vydělšení naivního uživatele a následné nabídce řešení a pomoci s odstraněním takto uměle vytvořeného problému pouhým objednáním licence na tento nechtěný software, jeví se jako jediným správným řešením podobná varovná hlášení ignorovat a takový program bez váhání odinstalovat ze systému, než napáchá nějaké skutečné škody.

3.2.3 Určení programu pro podrobnou bezpečnostní analýzu

Vzhledem ke zjištěným projevům chování programu Kristýna a jeho instalátoru, který nabízí mnoho podezřelých programů, a který z hlediska stanovených kritérií vyhověl nejlépe, jsem rozhodl vybrat ho jako vítěze výběrového řízení, který bude podroben dalšímu testování v bezpečnostní analýze, která bude obsahovat kontrolu síťové komunikace, pokud se program o nějakou pokouší, kontrolu stability programu a prověření případných nalezených podezřelých míst jako je například neočekávaný pád programu v reakci na uživatelský vstup.

3.3 Instalace programu

Po provedení všech potřebných příprav, které zahrnovaly zprovoznění podpůrného software pro virtualizaci, instalaci a nastavení virtuálního operačního systému spolu s instalací dalších programů potřebných pro účel práce, přišla řada na instalaci samotného vybraného zkoumaného software, který je předmětem této práce. S využitím datového úložiště, které zajišťovalo možnost přenosu dat mezi fyzickým strojem a virtuálním testovacím strojem jsem přenesl potřebný instalační soubor na virtuální pevný disk. Po zahájení monitorování chodu systému podle stanoveného postupu v analýze, který zahrnoval aktivaci záznamu v programu Process Monitor a spuštění programu Process Explorer z balíku SysInternals od Microsoftu a dalších monitorovacích nástrojů, zejména programu Wireshark, který zajišťoval záznam síťového provozu na místním síťovém adaptéru, jsem zahájil samotnou instalaci testovaného programu.

Při instalaci jsem simuloval chování běžného uživatele, který nejsnadnější možnou cestou provede průchod instalačním procesem a v průvodci instalací jsem proto v souladu se stanovenými postupy ponechal veškerá dostupná nastavení ve výchozích hodnotách. To samozřejmě vedlo k instalaci různého nevyžádaného software v podobě „užitečných“ nástrojů pro čištění počítače, vylepšení vyhledávání na internetu a zjednodušení práce s internetovým prohlížečem při použití doporučeného toolbaru.

Před samotnou instalací si instalační program vyžádal přítomnost balíčku .NET Framework, kterou nabídl ke stažení. Provedl jsem na žádost instalátoru stažení balíčku .NET Frameworkem a následně ho do systému nainstaloval, pak jsem opětovně spustil instalátor testovaného programu.

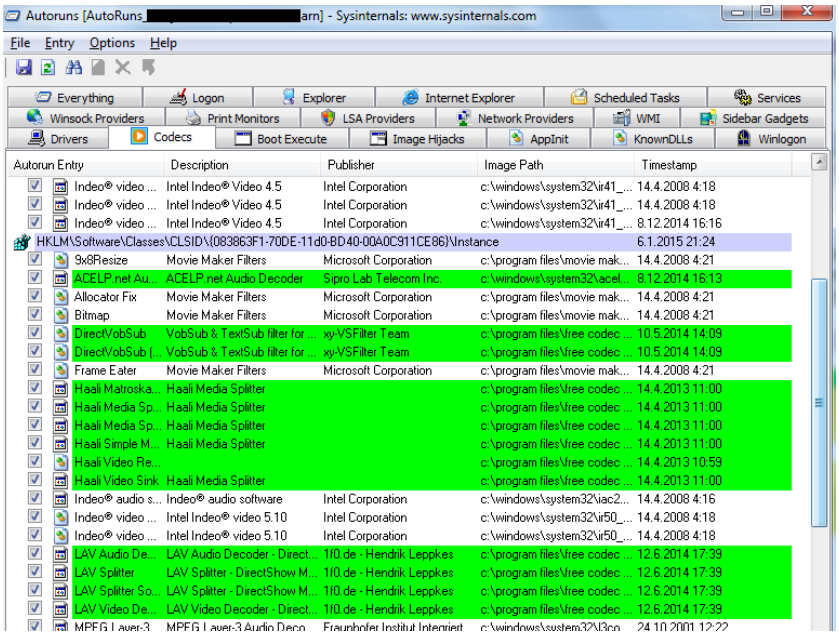
3.3.1 Kontrola změn v systému

Po proběhnutí instalace, která vzhledem k množství přidaného software trvala poměrně dlouho, jsem vyčkal dokončení instalačního programu, který ještě v závěru celého procesu spustil internetový prohlížeč se stránkou vydavatele software, která nabízela další programy a množství informací, které v tu chvíli nebyly příliš užitečné, a po skutečném dokončení celé instalace jsem zastavil monitoring činnosti systému, provedl jsem uložení získaných dat o celé události a přešel do další fáze průzkumu nasbíraných dat, abych se pokusil identifikovat důležité změny v systémových registrech týkající se jeho citlivých částí a také práci se síťovým rozhraním, které už při instalaci vykazovalo zjevnou aktivitu.

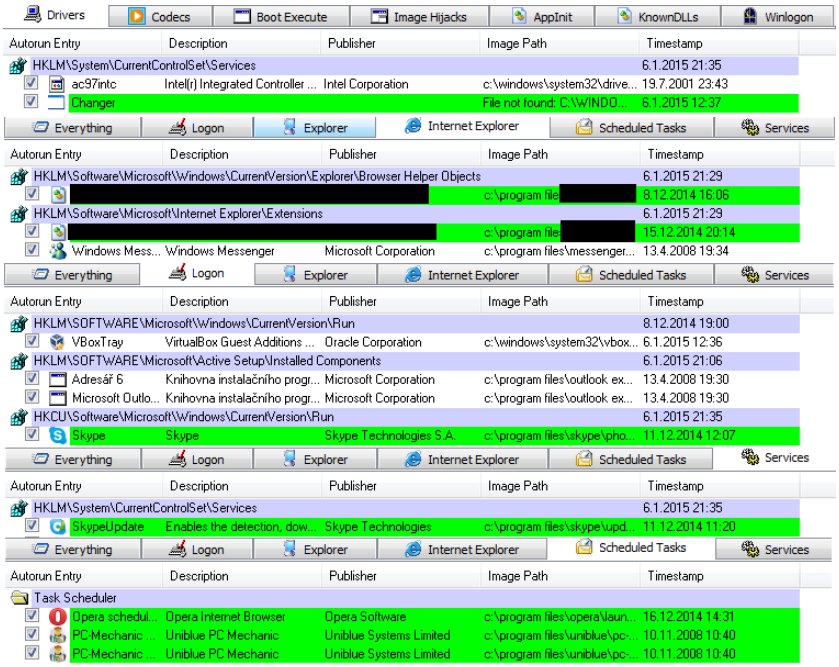
Při instalaci přistupoval instalátor do textového souboru, který vytvořil v dočasném adresáři uživatele a který zjevně slouží pro záznam informací o instalaci. Soubor obsahoval stručné údaje o operačním systému, čas spuštění instalace a téměř celý byl vyplněn údaji o instalovaných souborech. Na konci seznamu nainstalovaných souborů je záznam o registraci některých nainstalovaných knihoven (DLL) do systému příkazem RegSvr32.

3.3.2 Kontrola změn v registrech

V systémových registrech přibylo mnoho záznamů, které se ovšem po bližším prozkoumání týkaly přidavných nainstalovaných programů (Skype, Opera Internet Browser, PC Mechanic) případně se jednalo o záznamy aplikace Kristýna, které po startu spouštějí rozšíření pro webový prohlížeč Internet Explorer, jak je vidět z přiložených snímků obrazovek 3.2 a 3.3 z nástroje Autoruns.



Obrázek 3.2: Výsledky z programu autoruns



Obrázek 3.3: Výsledky z programu autoruns

3.3.3 Kontrola síťové komunikace při instalaci

Mnohem zajímavější se ukázaly být záznamy ze síťové komunikace, které jsem s pomocí programu Wireshark získal. Instalační program během celého instalačního procesu hojně komunikoval se servery, ze kterých stahoval informace v podobě XML dokumentů, obrázkových souborů a dalších dat. Zároveň se zjevně snažil i odesílat nějaká data o průběhu instalace. Vzhledem k množství odhalené síťové komunikace na pozadí jsem se rozhodl primárně zaměřit na její analýzu a odhalení případných slabin, které by mohly být zneužity pro útok na uživatele takového programu.

3.4 Hledání zranitelností v síťové komunikaci

Pro účely hledání zranitelností v síťové komunikaci programu byl proveden výběr zajímavých událostí ze záznamu o síťovém provozu při instalaci z programu Wireshark s použitím filtru podle typu protokolu, který byl po náhledu na počátek záznamu a objevení náznaků komunikace s použitím protokolu HTTP nastaven tak, aby došlo k zobrazení pouze těch záznamů, které představovaly HTTP požadavky na vzdálený server nebo odpovědi od dotazovaného serveru. Z takto vybraných záznamů byly vysledovány základní poznatky o komunikaci programu, podle kterých bylo rozhodnuto o směru dalšího postupu při rozboru nasbíraných dat.

3.4.1 Šifrování spojení

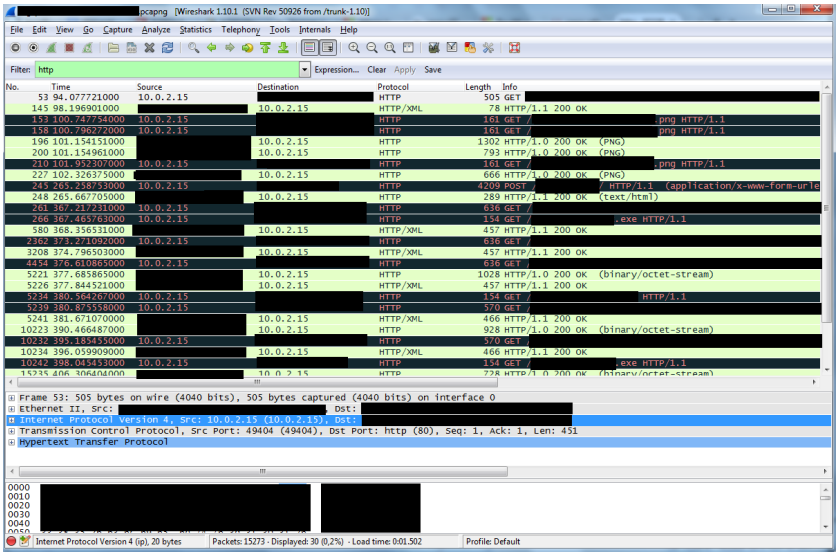
Z protokolu o komunikaci instalačního programu po síti vytvořeného nástrojem Wireshark bylo jasné, že komunikace je nešifrovaná. Jednotlivé packety bylo možné číst tak, jak byly zachyceny. To samo o sobě vytváří dojem, že data jsou buď nedůležitá a nedůvěrná resp. jejich případná neoprávněná změna či poškození při přenosu nemůže mít na běh instalačního procesu nebo na bezpečnost hostitelského systému a jeho uživatele žádný zásadní vliv nebo se jedná o potenciální bezpečnostní riziko, které by mohlo mít za následek narušení bezpečnosti nebo stability systému a mohlo by tak dojít ke způsobení nepříjemností nebo dokonce škod uživateli.

3.4.2 Analýza obsahu síťové komunikace

Byla provedena analýza zachycených dat, aby bylo zjištěno jaká data se přenášejí, zda-li je v pořádku, že nejsou šifrovaná či naopak se jedná o zranitelnost, která může útočníkovi snadno umožnit odposlech a krádež důvěrných informací, pokud se bude jednat o přenos údajů týkajících se operačního systému nebo jeho uživatelů. V případě, že by se jednalo o komunikaci mezi instalačním programem a vzdáleným serverem za účelem řízení instalace nebo analýzy jejího průběhu, bylo na místě prověřit možnosti narušení takového řídícího procesu a nalézt jeho slabá místa, u kterých by zneužití mohlo mít kritický dopad na celkovou úroveň zabezpečení.

3.4.3 Dynamické nabídky PUP

Při opětovném spuštění instalátoru na sytému ve výchozím stavu bylo aktivováno zachytávání síťové komunikace v programu Wireshark a celý průběh komunikace byl v návaznosti na akcích provedených v jednotlivých krocích instalačního průvodce zaznamenán. Jako první bylo zjištěno, že nabídky na přidavné (nevyžádané) programy, které instalátor nabízí uživateli, jsou dynamicky



Obrázek 3.4: Zachycení nabídky přídavných balíčků pro instalaci v programu Wireshark

získávány na začátku instalace ze vzdáleného serveru v podobě XML dokumentu, který obsahuje základní informace s celkovou nabídkou programů, které instalátor následné nabízí uživateli jako přídavek k instalaci původně staženého software. Část zachycené komunikace, která odpovídá této události je zachycena na obrázku 3.4 konkrétně na řádku 53, kde se nachází požadavek na XML dokument a pak na řádku 145, který obsahuje příchozí odpověď. Soubor XML s nabídkami obsahuje informace zkomprimované metodou gzip a zakódované do formátu base64. Ukázka jeho struktury je v následujícím výpisu.

```
Zachycený XML dokument:
<get_offers>
    <installer_md5s>
        <md5>0</md5>
    </installer_md5s>
    <session_key>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</session_key>
    <session_key>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</session_key>
    <session_key>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</session_key>
    <product_name>XXXXXXXXXXXX</product_name>
    <data encoding="gzip">...</data>
</get_offers>
```

Tag `data` obsahující zakódovaný obsah má atribut s názvem `encoding` a hodnotou `gzip`. To, že data jsou zakódovaná jako base64, bylo možné odhadnout z jejich formátu a informace o gzip kompresi byla získána z atributu encoding. Informace, že se opravdu jedná o gzip kompresi, se potvrdila po dekódování celé sekvence a získání čitelných dat v XML formátu. Dekódování těchto dat bylo pak už jen formálním krokem, který jsem realizoval za použití krátkého kódu v jazyce javascript s využitím knihovny `pako.js`, která poskytuje podporu pro operaci, kterou jsem potřeboval s daty provést. Jako výsledek jsem získal čitelná data s nabídkami přidavného software pro instalační program.

```
Kód pro dekódování komprimovaného obsahu XML dokumentu:
// zakódovaná zdrojová data
var b64Data     = '...';
// Dekódování base64 (konverze ascii > binary)
var strData     = atob(b64Data);
// Konverze binárního řetězce na pole čísel znaků
var charData    = strData.split('').map(function(x){
                                return x.charCodeAt(0);
                            });
// Konverze předchozího pole na pole znaků
var binData     = new Uint8Array(charData);
// Využití knihovny pako.js pro dekompresi
var data        = pako.inflate(binData);
// Převod dekomprimované informace do čitelného řetězce
var strData     = String.fromCharCode
                        .apply(null, new Uint16Array(data));
// Výtisk výsledku do konzole
console.log(strData);
```

Následující ukázka souboru XML zobrazuje jeho důležité části pro provedení modelového útoku. Zobrazena je jedna položka z kolekce nabídek přídavných balíčků k instalaci, která je vyjádřena značkou `offer`. Mezi důležité části ve vnitřní struktuře položky, kterými se budeme dále zabývat, patří obsah značek `package_url`, `package_md5` a `package_filesize`, které budou klíčové pro podvržení škodlivého instalačního balíku. Další důležitá značka má název `area` a pro zneužití nás bude zajímat zejména její atribut s názvem `gotourlonclick`.

```
Důležité části dekódovaného obsahu ze značky data:
<offer id="0000" instance_id="00000" language="en" advertiser_id="000">
        <name><![CDATA[XXXXXXXXX]]></name>
        <exclusions tags="cat_utility_registry"/>
        <filters applies="" keys=""/>
        <package_url>http://xxx.server.com/xxxxxxxx.exe</package_url>
        <package_md5>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</package_md5>
        <package_filesize>1000000</package_filesize>
        <package_tool_version/>
<styles>
    <image_source name="imgsrc">
        http://xxx.server.com/xxxxxxxx.png
    </image_source>
</styles>
<banner_title><![CDATA[XXXXX]]></banner_title>
<banner_description><![CDATA[XXXXX]]></banner_description>
<panel>
    <image name="background-on" imagesource="imgsrc"
        size="471,220" position="0,0" normalimagefrom="0,0"/>
    <panel id="1" name="install" Type="Radio" default="2"
        tabstopgroup="true" drawfocused="false">
        <radio id="2" name="yes" position="368,150" size="100x20" drawfocused="false"/>
        <radio id="3" name="no" position="368,176" size="100x20" drawfocused="false"/>
    </panel>
    <area position="10,185" size="140x10" gotourlonclick="http://www.server2.com/xxx"/>
    <area position="182,185" size="64x10" gotourlonclick="http://www.server2.com/xxx"/>
        <area position="10,185" size="140x10" gotourlonclick=""/>
        <area position="182,185" size="64x10" gotourlonclick=""/>
    </panel>
    <icon_url>http://xxx.server.com/xxxxxxx.ico</icon_url>
    <validation_code><![CDATA[xxx]]></validation_code>
</offer>
```

3.5 Zneužití XML dokumentu s nabídkou PUP

Při rozboru získaných dekomprimovaných dat jsem zjistil celou řadu velice závažných potenciálních bezpečnostních problémů, které mohou způsobit přímé ohrožení i zkušenějšího uživatele. V obsahu dokumentu se totiž vyskytují odkazy na zdroje pro instalátor, a to na balíčky software, ale i na obsah některých obrazovek v jednotlivých krocích. Instalátor se pokouší stahovat obrázky, které následně ukazuje uživateli jako další kroky instalace a zároveň na jednotlivých obrazovkách umožňuje zobrazit aktivní prvky jako jsou odkazy nebo formulářové prvky. Právě tyto odkazy by mohl potenciální útočník zneužít k napadení uživatele, který si bude chtít přečíst například licenční podmínky nebo další informace o nabízeném nevyžádaném software, na které odkaz má směřovat. Proto bylo rozhodnuto tuto možnost prověřit a pokusit se XML dokument pozměnit a takto upravený dokument podvrhnout instalačnímu procesu, abychom zjistili, jak bude instalační program na podvržený XML dokument reagovat, a zda-li obsahuje nějaké bezpečnostní procedury, které podvržený dokument rozeznají.

V dokumentu s nabídkou PUP jsou obsažené i odkazy na zdroje s příponou `.exe`, které zjevně tvoří samotné instalační balíky nabízeného přídavného software. Zde jsem viděl další významný bezpečnostní problém, který by mohl nastat, pokud by došlo k záměně stahovaného exe souboru určeného pro instalaci přídavného programu, protože v takovém případě by mohlo dojít ke spuštění naprosto libovolného kódu na uživatelské stanici a vzhledem k tomu, že instalační proces si při svém spuštění vyžádá zvýšená oprávnění pro svůj běh, mohly by vzniklé škody být skutečně velmi rozsáhlé.

3.6 Modelový útok na instalátor

Pro útok s využitím známých informací o možných zranitelnostech z předchozího průzkumu chování instalačního procesu se nejlépe hodí útok typu `man in the middle`[1]. K provedení tohoto typu útoku bylo zapotřebí vytvořit server, který by zajišťoval funkci prostředníka ovlivňujícího komunikaci mezi legitimními účastníky komunikace, kterými v tomto případě jsou instalační proces na uživatelské stanici a vzdálený server vydavatele software.

Pro účely implementace serveru jsem využil server `Apache` ve verzi 2.2.21 a skriptovací jazyk `PHP`[2] ve verzi 5.2.8. Server Apache spolu s interpretem skriptovacího jazyka PHP jsem vybral ze dvou důvodů. Prvním byla jeho dostupnost a možnosti jednoduše a bez zbytečně složitých konstrukcí provádět komunikaci po protokolu HTTP a druhým byla letitá osobní zkušenost s používáním tohoto softwaru.

Se serverem, který bude simulovat útočníkův server maskovaný jako legitimní server vydavatele software, bylo zapotřebí docílit přesměrování komunikace mezi instalátorem a tímto serverem. Typickou metodou pro tento účel je napadení DNS záznamů a zajištění chybných překladů doménových názvů na IP adresu. V takovém případě je pak počítačům, které zasílají požadavek na překlad adresy, zaslána odpověď s chybným obsahem a takový počítač na základě těchto chybných podvržených údajů přistupuje na server, který ve skutečnosti na adrese zadané nic netušícím uživatelem

[1] Metoda útoku, která je založena na skutečnosti, že dvě komunikující strany mají dojem, že komunikují po přímé lince, která mezi nimi je, ale skutečnost je taková, že útočník jejich komunikaci odposlouchává a provádí její úpravy tak, že zúčastněnné strany o útočníkovi neví [24].

[2] Personal Home Page resp. PHP Hypertext Preprocessor je skriptovací jazyk vytvořený pro programování webových aplikací. Zkratka PHP s jejím novějším významem je tzv. backronym neboť je v sobě obsažena [7].

nemá být. Útok je nazýván jako `DNS spoofing` [13]. V této práci jsem se snažil narušit komunikaci mezi virtuálním strojem, na kterém probíhala testovací instalace a mezi vnějším serverem. Využil jsem proto svůj fyzický stroj, který byl pro ten virtuální bránou pro připojení k internetu a simuloval jsem napadení DNS serveru s využitím souboru hosts ve Windows, který umožňuje definovat překlady adres na IP adresy, kterými se potom systém primárně řídí. Proto jsem do tohoto souboru umístěného v adresáři C:\Windows\System32\Drivers\etc na virtuálním stroji přidal záznamy o falešných překladech domén, na které směřují požadavky instalátoru, a nasměroval tak komunikaci na svůj fyzický stroj s běžícím serverem Apache, který se postaral o zprostředkování komunikace a její potřebné úpravy.

Důležité bylo zajistit, aby veškeré požadavky směřující na lokální server na fyzickém stroji byly správně obslouženy a proto jsem vytvořil pravidlo pro přesměrování všech požadavků na hlavní soubor s obslužným skriptem `index.php` a učinil jsem tak využitím konfigurace serveru přes speciální konfigurační soubor `.htaccess`. Hlavnímu oslužnému skriptu jsem v parametru předal vždy celou adresu, na kterou se klient dotazuje. Teprve v těle skriptu bylo rozhodnuto, jak se s požadavkem naloží a jaká bude vrácena odpověď.

```
.htaccess :
DirectoryIndex index.php
<IfModule mod_rewrite.c>
    RewriteEngine on
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule ^([^?]*)$ /index.php?path=$1 [NC,L,QSA]
</IfModule>
```

Aby nebylo nutné implementovat veškerou logiku, kterou disponuje skutečný a pravý server poskytovatele software, využil jsem poznatku, že samotná komunikace není nijak šifrována nebo jinak zabezpečená a tudíž je možné jí bez potíží z lokálního podvodného serveru směrovat na skutečný server a získanou odpověď předat zpět dotazujícímu se klientovi. Po úspěšném otestování funkčnosti pouhého přesměrovávání komunikace jsem započal s implementací procedur, které měly za úkol pozměnit komunikaci za účelem provedení útoku na uživatele resp. na instalační program.

3.6.1 Zneužití odkazů v instalačním průvodci

Po dokončení falešného serveru, který bude řídit komunikaci mezi napadeným klientem a původním serverem, bylo zapotřebí otestovat možnosti změny obsahu XML dokumentu s nabídkou přídavných instalačních balíčků, který je stahován ze serveru na začátku instalačního procesu. Účelem bylo zjistit, jestli instalační program tyto změny bude schopen odhalit a podvržený XML dokument odmítne, nebo bude pokračovat ve svém běhu i s nepravým dokumentem.

Ověření zabezpečení XML dokumentu proti změnám bylo spojeno s prvním útokem, který měl umožnit podvržení odkazů na škodlivé stránky přímo do instalačního průvodce s využitím zmíněných atributů s názvem `gotourlonclick` ve značce `area`. Pokud by instalační program neoprávněnou změnu v XML dokumentu dle předpokladů nezjistil, mohlo by dojít k zobrazení škodlivých odkazů na příslušných obrazovkách v průvodci instalací.

Na falešném serveru byl nasazen skript v jazyce PHP, který při přenosu klíčového XML dokumentu zachytil data ze serveru vydavatele programu a pozměnil v nich odkazy obsažené v atributech s názvem `gotourlonclick`. Bylo využito regulárních výrazů pro definici vzorků, které mají být

nahrazeny. Pro účely modelového útoku byly vloženy odkazy na neexistující stránky. Následující výpis ukazuje stěžejní část skriptu.

```
Kód pro náhradu odkazů v~XML dokumentu:
$cnt; // obsahuje načtenou odpověď ze skutečného serveru
// zde se provádí záměna odkazů
$cnt = preg_replace("/gotourlonclick\=\"http[s]?[^\"]+/",
      "gotourlonclick=\"http://www.worm.com", $cnt);
```

Při spuštění instalačního průvodce a kliknutí na odkaz na obrazovce v kroku, kde je nabízena instalace přídavných balíčků, bylo očekáváno přesměrování na neexistující webovou stránku. Ostatní požadavky nebyly ovlivněny a celá komunikace až na změnu v obsahu XML probíhala se skutečným serverem vydavatele. Falešný server pouze přeposílal požadavky a odpovědi mezi oběma stranami. Po spuštění instalačního procesu a průchodu úvodními kroky byla zobrazena výzva na instalaci nabízeného přídavného programu. Obrazovka obsahovala dva odkazy na podmínky použití a na informace o ochraně soukromí. Po kliknutí na každý z odkazů ale došlo k nasměrování uživatele na podvodné stránky s adresou, která byla podstrčena do XML dokumentu prostřednictvím PHP skriptu obsahujícího výše zmíněný kód.

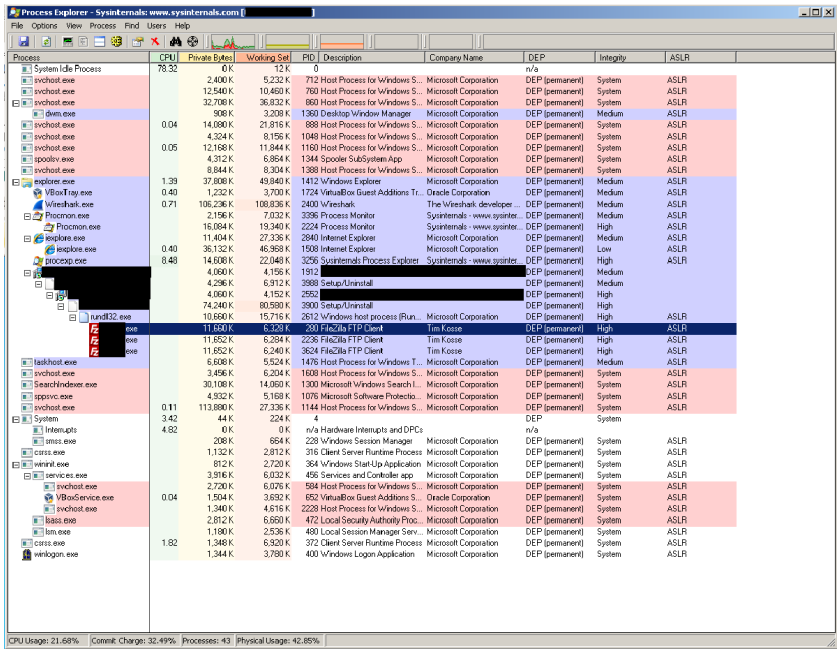
Tímto krokem bylo zjištěno, že XML dokument není nijak zabezpečen proti provádění změn a můžeme proto přistoupit k dalším testům, které by mohly potvrdit další obavy z předpokládaných zranitelností, a to je především záměna instalačních balíků s nabízeným přídavným software.

3.6.2 Záměna instalačních balíčků

V záznamu ze sledování komunikace programem Wireshark jsem nalezl požadavky, které jsou odesílány na server za účelem stažení souboru s příponou `.exe`. Požadavek na tento soubor jsem proto zachytil na falešném serveru a místo obsahu, který byl požadován ze skutečného serveru, jsem se snažil podstrčit instalátoru obsah jiného `exe` souboru. Po dokončení této úpravy skriptu zajišťujícího řízení komunikace na falešném serveru a otestování přijaté odpovědi, která skutečně obsahovala podvržený obsah jsem opětovně spustil instalační proces. Došlo k nabídnutí přídavného software, který jsem přijal a následně se spustila instalace. Ke spuštění podvrženého souboru ovšem nedošlo.

Započal jsem s vyhodnocováním pozorování, abych se pokusil zjistit, co zabránilo spuštění staženého instalačního balíku, který byl zaměněn za jiný potenciálně škodlivý. Ostatní přídavné programy se v pořádku nainstalovaly. Detailním průzkumem instalačního procesu jsem zjistil, že při potvrzení nabídky na instalaci přídavného software dochází ke spuštění dalšího procesu s názvem [redacted], který zjevně zajišťoval stažení instalačního balíku. Vyplývalo to z parametrů spuštění tohoto podprogramu, které obsahovaly adresu na stahovaný soubor. Soubor byl stažen, ale k jeho spuštění nedošlo, to bylo pro mě známkou přítomnosti nějakého kontrolního procesu v tomto podprogramu, který je schopen rozpoznat změnu instalačního balíku.

Znovu byl prověřen XML dokument, který obsahoval nabídky na tento přídavný software a bylo zjištěno, že jsou v něm přítomné značky s MD5 otisky a s dalšími informacemi o nabízeném souboru. Tím důležitým parametrem byla mimo MD5 otisku datová velikost souboru. Oba tyto údaje byly zmíněny v sekci 3.4.3. Byla proto provedena další úprava skriptu, který zajišťoval ovlivnění komunikace mezi serverem a klientem, a byla do něj přidána implementace nahrazování potřebných MD5 otisků a údajů o velikostech souboru údaji, které odpovídaly vlastnostem podvrženého potenciálně škodlivého souboru s příponou `.exe`. Celý průběh byl pozorně sledován s využitím zmíněných nástrojů ze Sysinternals Suite. Zejména nástroj Process Explorer se uplatnil při přehledném



Obrázek 3.5: Spuštění podvržené aplikace (pro účely testu byl použit FTP klient FileZilla)

zobrazování všech běžících procesů a ukazování změn, které se týkaly spouštění nových a zániku dokončených procesů. Při sledování průběhu instalačního procesu došlo k nastartování podprogramů určených pro stahování instalačních balíků s přidavným software z nabídky, které byly přeneseny ze serveru na klientskou stanici, kde po dokončení stažení a zřejmě i po ověření souhlasu MD5 otisku a velikosti souboru, došlo ke spuštění těchto podvržených `exe` souborů, a to dokonce se zvýšeným oprávněním zděděným od instalačního procesu, tudíž ani nebylo nutné, aby uživatel znovu potvrzoval přidělení zvýšených oprávnění těmto novým procesům.

3.6.3 Zhodnocení závažnosti zranitelností

Vzhledem k tomu, že se podařilo podvrhnout uživateli odkazy na škodlivé stránky a bylo zjištěno, že program až na ověření MD5 otisku a velikosti souboru neprovádí žádnou další kontrolu stahovaných exe souborů s nabízeným přídavným software, je zranitelnost obsažená v tomto instalačním programu velice závažná, neboť dovoluje útočníkovi zcela automaticky provést hromadný útok všude, kde se mu podaří otrávit DNS cache a způsobit přesměrování komunikace na jeho škodlivý server a v horším případě i nepozorovaně spustit škodlivý kód přímo na uživatelské stanici, a to s oprávněním administrátora.

Zranitelnost jsem neoznačil za kritickou jen z toho důvodu, že se nachází v instalačním procesu a útočník by musel útok provádět v okamžiku, kdy se uživatel rozhodne instalovat tento program. Není to nemožné a při realizaci automatického útoku na dostatečně široký okruh uživatelů třeba se zapojením metod sociálního inženýrství, kdy útočník rozešle například v internetové kavárně nebo v podnikové či školní síti e-mailové zprávy s neodolatelnou nabídkou na stažení vynikajícího užitečného software zdarma, může být útočník úspěšný, ale není to tak závažné jako kdyby stejná

zranitelnost byla i v samotném programu, který uživatel bude používat pravidelně. I přesto, že jsem zatím neprozkoumal chování samotného nainstalovaného programu, považuji software Kristýna za velice nepříjemný zdroj rizika.

3.7 Průzkum běhu programu

Po prozkoumání instalátoru a prověření zjištěných zranitelností jsem se zaměřil na kontrolu samotného zkoumaného programu. Postupně jsem prověřil činnost programu a zjistil následující informace.

3.7.1 Analýza knihoven

S využitím nástrojů pro získání seznamu používaných dynamických knihoven jsem se zaměřil především na využití ASLR, které má zásadní význam při zabezpečení aplikací z hlediska prevence proti útokům typu přetečení bufferu. Process Explorer poskytuje možnost zobrazit používané knihovny a u nich podat informaci o využití ASLR. Ze získaného výpisu jsem zjistil, že program používá některé knihovny, které ASLR nevyužívají. Jedná se o knihovnu zkoumaného programu s názvem [redacted], o čtyři knihovny z prostředí .NET Frameworku ve verzi 2 s názvy COR-PerfMonExt.dll, diasymreader.dll, mscorsec.dll a PerfCounter.dll. Poslední externí knihovnou nepoužívající ASLR je rozšířená zlib1.dll, která obsahuje implementaci některých kompresních algoritmů. Knihovna zlib1.dll je distribuována se zkoumaným programem ve verzi 1.2.8.0.

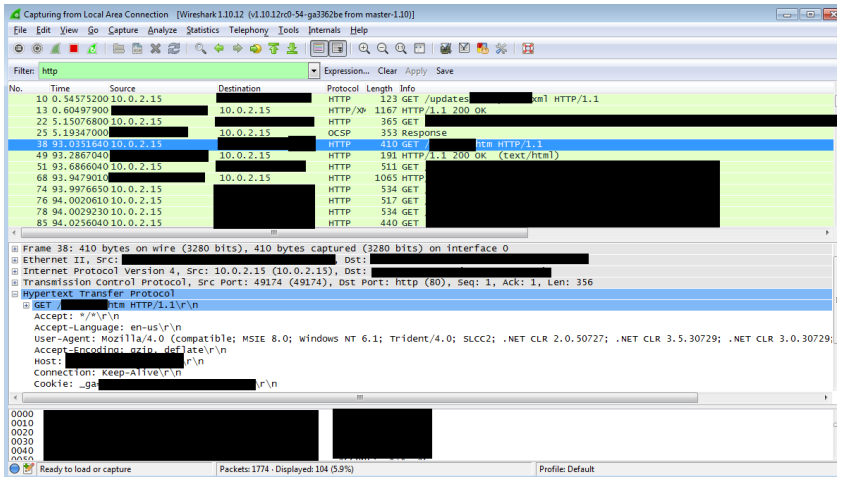
Používání knihoven, které nejsou přeloženy tak, aby používaly ASLR, je velice nebezpečné, protože použití ASLR u ostatních knihoven v podstatě ztrácí význam, neboť útočník vždy nalezne pevnou adresu v paměti směřující právě do knihovny, která nepoužívá ASLR, a to je pro něj pevný bod v aplikaci, který se stává ideálním cílem pro provedení útoku s využitím zranitelnosti typu buffer overflow.

3.7.2 Běžící procesy a služby

Spolu s hlavním programem se po spuštění nastartuje proces [redacted], který se po pár sekundách sám ukončí. Tento proces byl aktivní i při instalaci a staral se o stažení instalačních souborů. V tomto případě je zřejmě aktivován za účelem stahování aktualizací programu, protože tuto funkcionalitu program obsahuje a umožňuje tím uživateli pravidelně aktualizovat software na novější verzi, na kterou ho automaticky upozorní v hlavním okně programu.

3.8 Průzkum aktualizačního procesu

Aktualizační proces je sám o sobě běžná věc, na kterou jsou uživatelé zvyklí, ale z bezpečnostního hlediska představuje poměrně rizikovou funkcionalitu, která může způsobit při nesprávném zabezpečení, značné potíže s fatálními následky. Proto jsem se v souladu se zadáním práce a po zkušenostech s instalačním procesem, který značně snížil moji důvěru v tento software a jeho úroveň zabezpečení, zaměřil hlavně na prověření aktualizačního procesu, který je v programu obsažen.



Obrázek 3.6: Záznam požadavku na XML dokument pro aktualizaci

3.8.1 Šifrování komunikace

Pro zjištění základních infromací o aktualizačním procesu a o formátu probíhající komunikace při detekci nových verzí i při stahování aktuálních souborů, jsem využil opět program pro monitorování síťového provozu s názvem Wireshark. Po jeho aktivaci jsem spustil program a sledoval probíhající komunikaci, která probíhala podle zachytávaných dat opět nešifrovaně. Absence jakéhokoliv šifrování při průběhu aktualizace, kdy je uživatel upozorněn na novou verzi programu, která může být následně prostřednictvím uživatelského rozhraní tohoto software stažena do počítače a spuštěna, je přímo hazardem s bezpečností uživatele a jeho stroje. Falešné hlášení o aktualizaci může způsobit, že uživatel, který nemá při běžné práci prostředky k ověření pravosti komunikace a pokud není obzvláště opatrný a neprověřuje každý instalační balíček v sandboxu[3] nebo na testovacím virtuálním stroji, a to většina uživatelů nedělá, se stane obětí velice snadno a do jeho počítače může být nainstalován škodlivý software.

3.8.2 Metoda zjišťování nových verzí

V úvodu záznamu síťové komunikace programu jsem objevil HTTP požadavek na XML dokument. Odpověď na požadavek obsahovala zjevně seznam verzí softwaru, podle kterého program na uživatelské stanici vyhodnocuje, zda se má aktualizovat či nikoliv. Komunikace opět probíhala nešifrovaně (řádek 10 na obr. 3.6) bez jakéhokoliv zabezpečení a mohla být bez problému přečtena v podobě v jaké došlo k jejímu odposlechnutí. Jedná se zjevně o podobný případ jako v instalaci programu.

XML dokument obsahuje pouze seznam čísel verzí, které nijak logicky nenavazují, ale v případě, že znovu využijeme modelový útok typu `man in the middle`, bude s největší pravděpodobností možné změnit obsah souboru podle potřeby bez toho, aniž by zkoumaný program detekoval,

[3] v překladu pískoviště - tímto pojmem se označuje prostředí, ve kterém je program spuštěn, ale má značně omezená oprávnění a nemůže provádět invazivní změny v systému. Mnohdy je mu zabráněno provádět i změny na datových úložištích a v registru systému. Slouží pro izolování nedůvěryhodných programů, které uživatel chce spustit, ale plně jim nemůže důvěřovat [13].

že se jedná o neoprávněný zásah do přijímaných dat. Zkoumání změn obsahu a vlivu těchto změn na upozornění uživatele na novou verzi bude předmětem dalšího průzkumu v rámci podniknutí zkušebního útoku na aktualizační proces.

```
<Versions>
            <Version>6.4.2.113</Version>
            <Version>2.0.33.113</Version>
            <Version>5.0.55.113</Version>
            <Version>0.6.11.127</Version>
...
</Versions>
```

3.8.3 Analýza aktualizačního procesu

Při stahování instalačního souboru s aktualizovaným programem jsem zaznamenal skutečnost, že stahování probíhá paralelně ve více etapách, neboť v záznamu síťové komunikace byly vidět HTTP požadavky, na které přicházely od serveru odpovědi se stavovým kódem 206, který označuje odpověď typu „Partial content“, která, jak již název napovídá, obsahuje pouze část z nějakého celku.

Stahování aktuálních souborů začíná HTTP požadavkem typu HEAD na vzdálený server. Odpověď na tento požadavek je typicky pouze výpis informací z hlavičky shodný s informacemi v odpovědi na požadavek typu GET jen se v tomto případě neodesílá samotné tělo odpovědi tzn. data, na která se klient dotazuje nejsou odeslána. Následovala série několika požadavků typu GET, na které přišla od pravého serveru odpověď zmiňovaného typu „Partial content“ se stavovým kódem 206, který dle specifikace protokolu HTTP tento druh odpovědi od serveru označuje [2].

Při současném pozorování průběhu stahování podle indikátoru obsaženého v uživatelském rozhraní zkoumaného programu jsem sledoval i probíhající komunikaci v záznamu programu Wireshark. Mimo samotný přenos dat, která tvořila součásti nového instalátoru programu, už nedocházelo k dalším spojením za účelem řízení stahování či instalace. Po dokončení přenosu a získání kompletních odpovědí na zaslané HTTP požadavky typu GET směřující na server vydavatele softwaru, mělo dojít zřejmě ke spuštění aktualizačního procesu, to se ovšem nestalo a program místo toho při všech pokusech havaroval a byl ukončen operačním systémem, protože přestal reagovat.

Pro vyvolání dialogu s dotazem na uživatele za účelem spuštění stažení aktualizace bylo zapotřebí poskytnout programu XML dokument, o který v úvodu při kontrole nových verzí na vzdáleném serveru program žádá a který bude obsahovat takové číslo verze, aby se program nainstalovaný na klientské stanici domníval, že je zastaralý a potřebuje provést aktualizaci. Vzhledem k tomu, že struktura XML dokumentu byla velice strohá a neobsahovala kromě desítek řetězců v typickém formátu, který se používá pro označení verze programu, žádná další data, podle kterých by bylo možné odhadnout, které číslo verze je potřeba pozměnit na vyšší, rozhodl jsem se proto vyzkoušet postupně měnit každé jedno číslo verze až program zareaguje a zobrazí výzvu k aktualizaci.

3.8.4 Pád programu po stažení aktualizace

Po stažení aktuální verze programu, která byla stažena z původního pravého serveru vydavatele došlo k pádu programu, a to po každém pokusu o aktualizaci. Podrobil jsem proto program průzkumu na možnost výskytu zranitelnosti při čtení z neplatné paměti vlivem chybného řízení cyklu nebo dalších špatně ošetřených konstrukcí v programu s použitím ladícího nástroje OllyDBG, ve kterém jsem program krokoval ve chvíli, kdy mělo dojít ke spuštění aktualizace. Zjistil jsem, že dochází

k požadavku na přístup do paměti s nulovou adresou obsaženou v jednom z registrů použitého pro adresaci této paměti a na základě výskytu této neplatné operace je program ukončen.

Při zkoumání příčiny takového chování bylo přímo v disassemblovaného kódu aplikace hledáno, jaký by mohl být účel přístupu do neinicializované paměti a při diagnostice s použitím dostupných prostředků s využitím programu Dependency Walker bylo zjištěno, že existují závislosti na dynamicky načítaných knihovnách, které nelze v systému nalézt a jedná se o knihovny, které jsou součástí balíku `Microsoft Visual C++ 2012 Redistributable`. Bylo proto provedeno doinstalování tohoto softwarového balíku do testovacího systému běžícího na virtuálním stroji a následně došlo k dalšímu pokusu o spuštění aktualizačního procesu, který tentokrát proběhl korektně a došlo ke zobrazení dialogového okna s možností vybrat si spuštění instalace stažené aktualizace ihned nebo po ukončení práce s programem.

Je překvapivé, že program vykazuje toto podivné chování a vzhledem k tomu, že vyžaduje přítomnost těchto knihoven pro svůj běh, i když se to týká zjevně pouze vybraných funkcí programu jako byl proces aktualizace, mělo by být v instalačním procesu zobrazeno upozornění, které uživateli sdělí, že je potřeba příslušný balík podpůrných knihoven doinstalovat stejně jako tomu bylo v případě požadavku na doinstalování .NET Frameworku.

3.8.5 Úprava serveru pro útok na aktualizační proces

Pro útok na aktualizační proces jsem využil stejný server Apache s interpretem jazyka PHP běžící na lokálním stroji, na kterém byl spuštěn i virtuální testovací stroj s testovaným programem v prostředí Windows 7 ve 32bitové verzi. Samotnou logiku skriptu bylo ale potřeba upravit tak, aby se přizpůsobila formátu komunikace, která byla zachycena při zkoumání aktualizačního procesu a která se ukázala být odlišná od formátu komunikace použité pro stahování přídavných instalačních balíků v instalačním průvodci, kde se již podařilo úspěšně napadnout probíhající komunikaci a propašovat do cílového systému škodlivý kód, který byl následně spuštěn bez varování a se zvýšeným oprávněním.

Skript psaný v jazyce PHP řídící přesměrování komunikace mezi klientem v testovacím virtuálním stroji a vzdáleným serverem vydavatele software, který v rámci svého běhu ovlivňoval a měnil probíhající komunikaci za účelem provedení útoku na klientskou stanici, jsem upravil tak, aby byl schopen odpovídat na požadavky typu HEAD a aby dokázal reagovat i na požadavky typu GET se speciálním parametrem v hlavičce, který určoval rozsah požadované části ze souboru s aktuální verzí programu.

3.8.6 Ladění serveru na základě nově získaných dat

Po odladění drobných chyb a překlepů ve zdrojovém kódu ovládacího skriptu se podařilo podstrčit pozměněné XML zkoumanému programu a vzhledem k tomu, že soubor obsahoval řádově pouze desítky záznamů, byla proto zvolena metoda postupného provádění změn v jednotlivých řetězcích a postupně docházelo ke zvyšování číslel verzí v souboru až se podařilo po několika pokusech nalézt řetězec, u kterého změna na vyšší verzi přinutila program zobrazit výzvu k aktualizaci. Tento XML soubor byl zaměněn za původní odpověď od pravého serveru vydavatele software, aby v programu došlo při každém pokusu zjistit novou verzi k vyvolání nabídky k aktualizaci.

Po provedení změn v souboru, kterým lze řídit vyvolání upozornění na aktualizaci, následovalo podvržení škodlivého souboru s instalátorem nové verze potom, co uživatel klikne na výzvu k instalaci aktualizace. Při prvním pokusu o podvržení instalačního souboru se akce nezdařila. Podle kontroly záznamu v nástroji Wireshark docházelo sice k odesílání dat ze strany falešného serveru směrem ke klientovi, ale indikátor průběhu stahování v samotném programu nereagoval. Ovládací skript byl z tohoto důvodu znovu zkontrolován na přítomnost chyb a po vyloučení syntaktických chyb a výskytu varovných hlášení ze strany interpreteru PHP, bylo předpokládáno, že zřejmě došlo k logické chybě při řízení komunikace. Dalším krokem proto byla kontrola záznamů z nástrojů Wireshark a Process Monitor za účelem získání podrobnějších informací o práci programu a jeho interakci se vzdáleným serverem, který byl v tuto chvíli simulován falešným serverem, který komunikaci zprostředkovával a snažil se ji měnit za účelem provedení modelového útoku.

3.8.7 Nalezení interních záznamů aktualizačního procesu

Při procházení zachycených událostí v nástroji Process Explorer se ukázalo, že hlavní program při aktualizaci spouští proces s názvem [redacted], který se zjevně stará o stažení nového instalačního balíku. Tato informace vyplývala z tvaru příkazu, který byl programem vyvolán a obsahoval parametry specifikující adresu ke vzdálenému souboru a celočíselný parametr, jehož význam byl objasněn později v sekci 3.8.8. Důležitým zjištěním v tomto bodu rozboru aktualizačního procesu bylo nalezení souboru se záznamem o stahování aktualizací, který proces [redacted] vytvořil v dočasném adresáři uživatele umístěném v souborovém systému v adresáři s daty aplikací[4] a který umožňoval nahlédnout do postupu, jaký je užíván tímto procesem pro stažení souboru s aktualizací. Bylo provedeno odstavení falšeného serveru za účelem vytvoření záznamu instalace při komunikaci s pravým serverem vydavatele, který pak byl použit pro porovnání se záznamem vzniklým při stahování aktualizace s aktivovaným falešným serverem a přesměrovanou komunikací. Výsledkem byla sada ladících a informačních výpisů, které program poskytl do záznamu a na základě kterých se ukázalo při vzájemném porovnání několik zajímavých odlišností.

```
Část interního záznamu programu o aktualizaci:
[18:05:49.62][0b5c][DBG][x] Start xxxxxxxx
[18:05:49.62][0b5c][DBG][x] param = 0 , value = xxxx
[18:05:49.62][0b5c][DBG][x] param = 1 , value = -t
[18:05:49.62][0b5c][DBG][x] param = 2 , value = download
[18:05:49.62][0b5c][DBG][x] param = 3 , value = -n
[18:05:49.62][0b5c][DBG][x] param = 4 , value = -i
[18:05:49.62][0b5c][DBG][x] param = 5 , value = 0
[18:05:49.62][0b5c][DBG][x] param = 6 , value = -p
[18:05:49.62][0b5c][DBG][x] param = 7 , value = xxx.exe
[18:05:49.62][0b5c][INF][x] --> downloadFile
[18:05:49.62][0b5c][INF][y] --> y_setDownloadThreadCount | count = 10
[18:05:49.62][0b5c][INF][y] ==> y_getRemoteFileSize | http://xxx.server.com/xxx.exe
[18:05:49.64][008c][INF][a] id=1 | nAttemptCount = 1
[18:05:50.19][008c][INF][a] id=1 download thread finished
[18:05:50.22][0b5c][INF][b] y_getRemoteFileSize | size = 10000000
[18:05:50.22][0b5c][INF][y] --> y_downloadToFile | http://xxx.server.com/xxx.exe | x
[18:05:50.27][03f8][INF][a] id=1 | nAttemptCount = 1
[18:05:50.79][03f8][INF][a] id=1 download thread finished
[18:05:50.81][0b5c][INF][b] y_getRemoteFileSize | size = 64578440
[18:05:50.81][0b5c][INF][g] G::readFile() | File does not exist: xxx.txt
[18:05:51.11][0b5c][INF][x] progress=0
[18:05:51.14][0e04][INF][b] ==> writeThread
```

[4] C:/Users/xxxx/AppData/Roaming/xxxxxxxxxx/logs

```
[18:05:51.14][0710][INF][a] id=100 | nAttemptCount = 1
...
[18:05:51.43][089c][INF][a] id=109 | nAttemptCount = 1
[18:05:51.52][0b5c][INF][x] progress=0
...
[18:05:53.46][0b5c][INF][x] progress=2.53438
...
```

3.8.8 Využití interních záznamů aktualizačního procesu

Bylo zjištěno, že program při svém spuštění nejdříve zasílá HTTP požadavky typu HEAD, které mají zřejmě za úkol získat od serveru přesnou velikost souboru a následně se provede rozdělení souboru do částí, jejichž počet je specifikován celočíselným parametrem. Program následně provede požadavky typu GET, na které očekává odpověďi se stavovým kódem 206 „Partial content“ a které zjevně tvoří tyto části , které při sjednocení vytvoří kompletní instalační soubor.

Při kontrole záznamu z aktualizace s aktivovaným falšeným serverem se hned v úvodu záznam odlišoval, a to po provedení dotazu na velikost souboru s aktualizací (getRemoteFileSize), na který zřejmě přišla neplatná odpověď, protože v záznamu byly uvedeny následující řádky. Na šestém řádku se nachází informace naznačující výskyt chyby (curl err = 7).

```
Část interního záznamu programu při připojení na falešný server:
...
[18:10:17.40][0ad8][DBG][x] param = 7 , value = xxx.exe
[18:10:17.40][0ad8][INF][x] --> downloadFile
[18:10:17.40][0ad8][INF][y] --> y_setDownloadThreadCount | count = 10
[18:10:17.40][0ad8][INF][y] ==> y_getRemoteFileSize | http://xxx.server.com/xxx.exe
[18:10:17.43][0eb0][INF][a] id=1 | nAttemptCount = 1
[18:10:38.68][0eb0][DBG][a] id=1 | RESPONSE_CODE: 0 | curl err = 7
[18:10:38.69][0eb0][INF][a] id=1 download thread finished
[18:10:38.80][0ad8][INF][b] getRemoteFileSize | while
[18:10:38.89][0c94][INF][a] id=10 | nAttemptCount = 1
[18:10:38.99][0ad8][INF][b] getRemoteFileSize | while
[18:10:39.09][0ad8][INF][b] getRemoteFileSize | while
[18:10:39.20][0ad8][INF][b] getRemoteFileSize | while
...
```

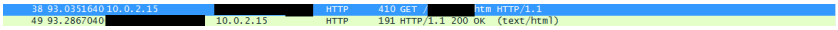
Název serveru je smyšlený, aby bylo zachováno utajení skutečného názvu aplikace z důvodu zabránění zneužití této práce jako návodu pro provedení útoku.

Na základě tohoto poznatku jsem zjistil, že zřejmě není v pořádku odpověď serveru, která má poskytnout informaci o velikosti stahovaného souboru, a proto jsem započal s hledáním příčiny v ovládacím skriptu v části zajišťující vytváření odpovědí na HEAD požadavky. Při procházení kódu skriptu jsem zjistil, že údaje odesílané v hlavičce odpovědi, které určují délku obsahu[5], chybí a tudíž program, který řídí aktualizaci, nedostane potřebné informace pro zahájení stahování a podle dalších záznamů o aktualizaci obsahujících informaci o získávání velikosti souboru, které se stále opakují, se zjevně snaží tyto údaje v pravidelných intervalech získat a v tomto bodě se zacyklí.

3.8.9 Úspěšné provedení útoku

Na základě zjištěných problémů byla provedena oprava logiky řídícího skriptu a tím bylo zajištěno, aby odpověď obsahovala potřebné údaje pro aktualizační program. Po nasazení nového aktualizovaného skriptu na server a spuštění aktualizačního procesu došlo k zahájení stahování. Stažení souboru

[5] délka obsahu se udává v bytech a odesílá se v hlavičce s názvem „Content length“



Obrázek 3.7: Záznam požadavku na stránku s výzvou k platbě

s aktualizací se zdařilo a hlavní program následně zobrazil výzvu uživateli v dialogovém okně, které nabízelo možnost aktualizovat program okamžitě, nebo naplánovat aktualizaci po ukončení práce. Po spuštění aktualizace došlo k nastartování podvržené aplikace místo toho, aby bylo detekováno podvržení neplatného souboru. Spuštěním podvržené aplikace bylo ukázáno, že je možné zneužít s využitím znalostí a postupů z provedeného rozboru tuto funkci aktualizace programu a popsaným způsobem ovládnout hostitelský systém, který může útočník využít pro své potřeby podle libosti a záleží jen na tom, jaké budou jeho úmysly, případně jak důmyslný a nenápadný bude jeho škodlivý software.

Vzhledem k tomu, že k provedení útoku stačilo jen přesměrovat komunikaci mezi serverem vydavatele a klientem a šikovným způsobem podvrhnout v inkriminovanou chvíli odlišný obsah stahovaného instalačního souboru, považuji tuto zranitelnost za vysoce nebezpečnou až kritickou, protože v případě jejího zneužití mohou být důsledky fatální. Přítomnost této zranitelnosti ukazuje, že při vývoji aplikace zřejmě nebyly zohledněny ani základní bezpečnostní zásady.

3.9 Prověření zabezpečení plateb v aplikaci

V aplikaci je možnost provádět online platby za účelem získání členství, které je označeno jako „Premium account“. Transakci lze provést přes známý platební systém PayPal. Analýza zabezpečení této transakce byla považována za nezbytnou, jelikož se jedná o funkcionalitu která operuje s penězi a je zde možné riziko odcizení převáděných finančních prostředků nebo přihlašovacích údajů k platební bráně, které by mohlo ohrozit uživatele a způsobit mu v případě výskytu nějakého incidentu finanční škodu.

3.9.1 Kontrola komunikace a šifrování

Jako první bylo aktivováno monitorování sítě s pomocí nástroje Wireshark, abychom získali záznam komunikace, která probíhá mezi klientem a serverem v době přesměrování k platební bráně pro platbu prémiového členství. Získaný záznam byl předmětem zkoumání, při kterém vyšlo najevo, že komunikace je zcela nešifrovaná (označený řádek č. 38), a tudíž je možné ji stejně jako v případě aktualizace odposlouchávat a zřejmě i měnit. V tomto případě dochází k nasměrování uživatele na stránku vydavatele software, kde jsou informace o platbě spolu s tlačítkem, které uživatele přesměruje na platební bránu. Přesměrování na platební bránu již probíhá s využitím protokolu HTTPS[6].

3.9.2 Nalezená zranitelnost a popis jejího zneužití

Problém je ovšem právě v připojení uživatele na nezabezpečenou stránku vydavatele softwaru, protože právě v tomto okamžiku lze uživateli podstrčit jinou stránku stejně jako bylo provedeno podstrčení falešného XML dokumentu řídícího zjišťování dostupnosti nových verzí programu, a z uživatele se tak může stát oběť, ze které útočník vyláká platbu za prémiové členství, které ale nikdy

[6]Zabezpečené spojení používající protokol HTTP s využitím SSL/TLS při přenosu dat na standardním portu 443 [22].

nebude uživateli poskytnuto, a zároveň může dojít i k odcizení přístupových údajů k platební bráně, pokud by si uživatel nevšiml, že z podvržené stránky je sice přesměrován do platební brány, ale pouze zdánlivě, neboť se může jednat o další podvrženou stránku, kterou by uživatel rozeznal jen podle bezpečnostního certifikátu a toho si běžný uživatel nemusí všimnout. Vzhledem k tomu, že se jedná o přístup do platební brány, je zde veliké riziko odcizení vysoce důvěrných osobních údajů, záznamů o transakcích a hlavně možnost páchání trestné činností s využitím tohoto účtu, kdy přes něj lze realizovat převody peněz pocházející například z kriminální činnosti.

3.10 Zabezpečení proti nestandardním vstupům

Po kontrole a rozboru síťové komunikace byly analyzovány reakce programu na nestandardní vstupy. Program na vstup přijímá ze systémové schránky řetězec, ve kterém se očekává hypertextový odkaz na zdrojový soubor umístěný na serveru [redacted]. Proto byl tento vstupní kanál v programu otestován na přítomnost zranitelnosti.

3.10.1 Testování reakce na příliš dlouhý vstup

Do systémové schránky byl umístěn text o délce přes 13000 znaků, který nebyl zcela náhodný, protože začínal validním podřetězcem, který měl formát odpovídající tvaru odkazu na soubor umístěné na serveru [redacted]. Dále odkaz obsahoval hodnotu parametru, která byla velice dlouhým řetězcem stejných znaků. Stejný vybraný znak byl použit za účelem detekce míst v paměti, na kterých by se případným zkopírováním takto dlouhého řetězce vyskytla nápadná místa s bloky stejných bajtů a podle toho by bylo možné určit, kde se nachází začátek bufferu pro čtení vstupu a zároveň detekovat případné přetečení tohoto bufferu s pomocí nástrojů pro výpis paměti, pokud by v programu situace nebyla správně ošetřena.

3.10.2 Zjištění podezřelého místa v programu

Program po zadání takto dlouhého řetězce na vstup provedl neplatnou operaci a byl operačním systémem ukončen. To bylo prvním signálem k tomu, že by zde mohla zranitelnost existovat, protože se mohlo stát, že došlo k přepisu některé z návratových adres na zásobníku a při ukončování podprogramu byl proveden skok do neplatné části paměti, a proto operační systém program ukončil.

3.10.3 Prověření potenciální zranitelnosti

Byl proveden start programu v ladícím nástroji OllyDBG, aby mohla být sledována jeho činnost během vkládání testovacího nadměrně dlouhého řetězce na vstup. Po inicializaci programu a načtení všech potřebných modulů došlo k vložení připraveného testovacího řetězce na vstup programu zvolením příslušného příkazu v nabídce hlavního menu. Následovalo čekání na zastavení programu. Při sledování běhu programu bylo zjištěno, že při spuštění této funkce stahování aktualizace se provádí kód přítomný v knihovně [redacted]. Proto byl do této knihovny umístěn bod přerušení, aby se vykonávání programu zastavilo ještě před jeho pádem a mohlo tak při krokování být zjištěno více podrobných informací o vnitřním stavu programu a jeho změnách, které předcházely zaznamenanému pádu.

Při krokování programu bylo objeveno několik míst, které obsahovaly kód pro detekci délky vstupu a jeho následného zkopírování do paměti. Délka řetězce se zjistila správně a došlo následně k volání funkce `malloc`, která na vstup dostala platné argumenty a provedla korektně alokaci paměti. Pro kontrolu bylo provedeno ověření délky řetězce, která byla 13246 znaků a registr `ECX` obsahoval po výpočtu délky hodnotu `33BE`, která tomuto číslu odpovídá.

```
Část kódu pro test délky řetězce:
:xxxxx.69FDF550
8A01; MOV AL, BYTE PTR DS:[ECX]
41;   INC ECX
84C0; TEST AL,AL
75F9; JNZ SHORT xxxxx.69FDF550
...
// EDX je počáteční adresa, ECX obsahuje konec bufferu
2BCA; SUB ECX, EDX
// nyní ECX obsahuje délku řetězce
```

Následně došlo v dalších krocích programu k volání funkce strcpy_s, která vyvolala výjimku a způsobila pád programu, jelikož tato výjimka nebyla zřejmě zachycena. Tato výjimka byla způsobena nastavením maximální velikosti kopírovaných dat, která byla zřejmě staticky zakomponována do programu a obsahovala hodnotu 4000. Podle rozboru funkce v disassemblovaném kódu se totiž prováděly následující akce:

Dojde ke kontrole platnosti všech 3 parametrů. Ověření spočívá v testu, který zkouší, zda není hodnota parametru rovna nule. První parametr je ukazatel na cílovou paměť, druhý udává maximální velikost pro bezpečné kopírování do cílového bufferu a třetím parametrem je ukazatel na zdrojový buffer.

```
Kód metody strcpy_s:
6D9947CD: 55                  PUSH EBP
6D9947CE: 8BEC                MOV EBP,ESP
6D9947D0: 56                  PUSH ESI
6D9947D1: 8B75 08             MOV ESX, DWORD PTR SS:[EBP+8]
6D9947D4: 85F6                TEST ESI, ESI
6D9947D6: 74 32               JE SHORT MSVCR120.6D99480A
6D9947D8: 8B55 0C             MOV EDX, DWORD PTR SS:[EBP+C]
6D9947DB: 85D2                TEST EDX, EDX
6D9947DD: 74 2B               JE SHORT MSVCR120.6D99480A
6D9947DF: 8B4D 10             MOV ECX, DWORD PTR SS:[EBP+10]
6D9947E2: 85C9                TEST ECX, ECX
6D9947E4: 0F84 D71A0400       JE MSVCR120.6D9D62C1
6D9947EA: 57                  PUSH EDI
6D9947EB: 8BFE                MOV EDI, ESI
6D9947ED: 2BF9                SUB EDI, ECX
6D9947EF: 8A01                MOV AL, BYTE PTR DS:[ECX]
6D9947F1: 88040F              MOV BYTE PTR DS:[EDI+ECX], AL
6D9947F4: 41                  INC ECX
6D9947F5: 84C0                TEST AL,AL
6D9947F7: 74 03               JE SHORT MSVCR120.6D9947FC
6D9947F9: 4A                  DEC EDX
6D9947FA: 75 F3               JNZ SHORT MSVCR120.6D9947EF
6D9947FC: 5F                  POP EDI
6D9947FD: 85D2                TEST EDX, EDX
6D9947FF: 0F84 D41A0400       JE MSVCR120.6D9D62D9
6D994805: 33C0                XOR EAX, EAX
6D994807: 5E                  POP ESI
6D994808: 5D                  POP EBP
6D994809: C3                  RETN
6D99480A: E8 6ECFFFFF         CALL MSVCR120._errno
```

Ve výpisu disassemblovaného kódu funkce, která vyvolává pád programu při zadání nadměrně dlouhého řetězce, který překročí zjištěnou délku 4000 znaků, je vidět úvodní vytvoření rámce na zásobníku v kódu na adrese `6D9947CD` a `6D9947CE`, kde dojde k záloze `EBP`[7] a k nastavení nové hodnoty na aktuální adresu uloženou v registru `ESP`[8]. Bezprostředně potom proběhne záloha registru ESI na zásobník. Tato zálohovaná hodnota je do registru zpět nahrána při ukončování kódu funkce na řádku s adresou `6D994807`, kde se nachází instrukce `POP ESI`. Začátek funkce po úvodní proceduře obsahuje kód na kontrolu parametrů, kde vždy trojice řádků od adresy `6D9947D1` obsahuje kontrolu parametru předaného funkci při volání na přítomnost nulové hodnoty. V případě, že je některá z hodnot nulová, program přeskočí na adresu `6D99480A`, kde dojde k vyvolání obsluhy této chyby. Kontrola je realizována nahráním hodnoty parametru ze zásobníku do registru instrukcí `MOV`, po které následuje instrukce `TEST`, která provede nastavení příznaků, které následně řídí provedení podmíněného skoku `JE SHORT` na adresu volání metody `MSVCR120._errno` zajišťující obsluhu chyby. Obsluha chyby je stejná pro první dvě kontroly (cílová paměť a maximální velikost). Třetí parametr (zdrojová paměť) je v případě chyby obsloužen na jiné adrese. V tomto konkrétním případě je to adresa `MSVCR120.6D9D62C1`.[9]

Po kontrolách parametrů dojde k záloze registru EDI na zásobník. Záloha se obnovuje na adrese `6D9947FC`. Následující operace zajišťuje přípravu pro jednodušší a efektivnější adresaci zdrojové a cílové paměti při kopírování dat. Na adresách `6D9947EB` a `6D9947ED` dochází k výpočtu, který do registru `EDI` umístí adresu cílové paměti zmenšenou o adresu zdroje. Tato operace má význam při řízení cyklu, kdy na řádku `6D9947F4` dochází k inkrementaci pouze jedné řídící proměnné `INC ECX` a díky předešlé úpravě ukazatelů není potřeba dvou řídících proměnných a dvou operací inkrementace. Tato operace provádí posun ukazatele na zdrojovou paměť, který je posunut na další znak a tím je připravena adresa pro kopírování dalšího bajtu. Adresace cíle se pak díky předchozímu zmenšení o počáteční adresu zdroje provádí pouhým součtem těchto registrů `[EDI + ECX]`, jak je vidět na řádku s adresou `6D9947F1`.

Instrukce `TEST AL, AL` provádí test na konec kopírovaného řetězce, který je označen nulovým bajtem. Pokud dojde ukazatel na konec řetězce (tzn. že hodnota registru AL, kam se načítá kopírovaný bajt, bude nulová), provede se skok na adresu `6D9947FC`, kde dojde k obnově registru `EDI` a hlavně ke kontrole, zda došlo k využití maximální povolené velikosti paměti, která byla na začátku načtena do registru `EDX` a na řádku s adresou `6D9947F9` docházelo při každém průchodu kopírovacím cyklem k její dekrementaci (`DEC EDX`). V případě, že je v registru `EDX` hodnota nula, běh programu je směrován na adresu `MSVCR120.6D9D62D9`, kde pokračuje vyhodnocení této výjimky. Pokud k tomuto stavu nedojde, funkce skončí normálně vynulováním registru `EAX`, obnovou registru `ESI` a návratem z funkce, který zruší vytvořený rámec zásobníku obnovou `EBP` a zavoláním instrukce `RETN`, která provede skok na návratovou adresu. Pojistka proti možnému zápisu za konec inicializované cílové paměti je na řádku s adresou `6D9947FA`. Zde totiž probíhá test podle příznaku nastaveného instrukcí `DEC EDX` a v případě, že výsledek této operace je nulový, znamená to, že byla překročena maximální velikost, další iterace kopírovacího cyklu již neproběhne a v závěru funkce se vyhodnotí vzniklá situace, která vede k vyvolání výjimky.

[7] Registr označující spodní hranici zásobníku (Base Pointer)

[8] Aktuální vrchol zásobníku (Stack Pointer)

[9] Adresy se mění s každým spuštěním programu, neboť tato knihovna byla kompilována pro použití s ASLR.

3.10.4 Ověření hraničního případu

Funkce `strcpy_s` je bezpečnou verzí starší funkce `strcpy`, která neobsahovala parametr omezující maximální délku kopírovaných dat a pokud je správně použita, zabraňuje vzniku zranitelnosti typu buffer overflow. Je zde ovšem možnost, že programátor může této funkci předat chybný parametr zejména v případech, kdy je paměť alokována dynamicky a její velikost není dopředu známá a je zjištěna až za běhu programu. Zde může dojít k předání chybné velikosti a vzniku zmíněného druhu zranitelnosti pokud by inicializovaná paměť měla menší velikost než je udávaná maximální velikost. Proto byl otestován i hraniční případ, kdy bylo zjištěno, že bez návaznosti na skutečnou délku řetězce se do funkce pro kopírování předává vždy hodnota `0xFA0`, která v desítkové soustavě odpovídá hodnotě 4000 a která je nahrána do registru `EDX` pro řízení ochrany proti zápisu za hranici inicializované paměti, a na základě toho jsem se pokusil vložit na vstup programu řetězec dlouhý přesně 4000 znaků a 3999 znaků, abych mohl pozorovat chování programu v těchto krajních situacích. Program zareagoval dle očekávání a spadl při překročení délky 3999 znaků na vstupu při vyvolání výjimky, která vznikla ve funkci strcpy_s z důvodu pokusu o překročení maximální velikosti cílového bufferu. Nedošlo proto k zápisu ani jednoho bajtu za hranici inicializované paměti. Ovšem vyvolaná výjimka nebyla v programu zjevně nijak ošetřena, a proto se operační systém postaral o ukončení programu.
`Hlavička funkce strcpy_s` [18]:

```
errno_t strcpy_s(
   char *strDestination,
   size_t numberOfElements,
   const char *strSource
);
```

3.10.5 Vyhodnocení rozboru pádu programu

Po zjištění všech podrobností, které byly popsány v předchozích odstavcích jsem dospěl k závěru, že v případě nestability programu při zadání nadměrně dlouhého řetězce, jehož hraniční délka byla definována programem na 4000 znaků, dojde k pádu, ale jednoznačná zranitelnost očekávaného typu přetečení bufferu se na tomto místě nevyskytuje, neboť cílová paměť byla správně inicializována na maximální velikost, která se shodovala s velikostí předané funkci strcpy_s a která zajišťuje zastavení kopírování v případě pokusu o zápis za hranici definované paměti. Rozhodně by bylo na místě takovou výjimku ošetřit a zajistit korektní zotavení nebo alespoň ukončení programu. Funkce strcpy_s správně vyvolala výjimku, ale vzhledem k tomu, že výjimka nebyla programem zachycena a její obsluhu zajistil až operační systém, došlo tím k zastavení a ukončení programu.

3.10.6 Shrnutí výsledků

Při bezpečnostní analýze bylo nalezeno několik vážných zranitelností. Většina byla způsobena nedostatečným zabezpečením síťové komunikace. Pro účely ověření jejich závažnosti byly provedeny modelové útoky, které měly za úkol prokázat zneužitelnost nalezených zranitelností, nebo ji vyvrátit. V případě zranitelnosti, která byla objevena v procesu stahování dodatečných instalačních balíčků v rámci instalace programu, bylo potvrzeno, že útočník může spustit v napadeném zařízení libovolný kód. Se síťovou komunikací při instalačním procesu souvisela i další zranitelnost, která umožňuje zneužít XML dokument s nabídkou přídavných instalačních balíčků a umístit do něj odkazy na podvodné webové stránky. Tyto odkazy budou uživateli zobrazeny přímo v instalačním průvodci. Při

spuštění programu byla objevena zranitelnost v procesu, který souvisí s nákupem přímo v aplikaci. Útočník může tuto chybu zneužít a vylákat z uživatele převod finančních prostředků, případně i získat přihlašovací údaje k platební bráně. Aktualizační proces obsahuje podobnou zranitelnost jako instalační proces a umožňuje stáhnout a spustit podvržený instalační balíček, který může obsahovat škodlivý kód. Poslední zjištěná nesrovnalost v podobě nestability programu při zadání příliš dlouhého vstupu byla prověřena a jednalo se o neošetřenou výjimku, která vznikla při pokusu kopírovat nadměrně dlouhý řetězec do paměti inicializované na fixní velikost. V tomto případě zafungovala ochrana implementovaná ve funkci strcpy_s, která zabránila zápisu mimo inicializovanou paměť a v tomto místě v programu nedošlo ke vzniku zranitelnosti typu přetečení bufferu.

Kapitola 4

Návrh bezpečnostních opatření

Na základě nalezených zranitelností v zabezpečení zkoumané aplikace byl proveden návrh vhodných opatření, která by měla změnšit nebo odstranit vzniklá rizika a tím zlepšit ochranu uživatele této aplikace případně i dalších podobných aplikací, neboť některé z uvedených doporučení platí i v obecné rovině.

4.1 Důvěryhodnost zdrojů software

Při stahování aplikací z internetu, které jsou volně dostupné na různých specializovaných serverech a k tomu jsou zdarma, je potřeba dbát nejvyšší opatrnosti, neboť takového softwaru je nepřeberné množství a často bývá napsán různými autory, kteří nemají potřebné znalosti a zkušenosti a tomu také odpovídá výsledná kvalita. Takové programy bývají často nestabilní, mohou působit problémy související s kompatibilitou s operačním systémem a jeho součástmi včetně dalšího nainstaloveného softwaru. Na podobné programy nejsou obvykle ani poskytovány žádné záruky a když uvážíme, že málokdy známe podrobné reference na samotného vydavatele, je používání programů získaného z těchto zdrojů značným rizikem.

4.2 Nedůvěřovat serverovým opatřením proti nákaze

Na velkých portálech, které se zaměřují na distribuci těchto programů, obvykle může uživatel najít informaci, která říká, že server je zabezpečen antivirovým programem poskytnutým pro účely ochrany návštěvníků a veškerý obsah, který je na serveru nabízen prošel kontrolou na přítomnost škodlivého software. Z vlastní zkušenosti ovšem vím, že se takovému prohlášení nedá důvěřovat, protože tu jsou hned dva základní problémy:

- První problém spočívá v nedostupnosti kvalitních a spolehlivých informací o tom, kdy a hlavně jak často jsou soubory kontrolovány, případně se jedná o neurčité a obecné informace, které nic konkrétního o kvalitě kontrol nevypovídají.
- Druhým problémem jsou samotné instalační balíky, které mohou obsahovat instrukce ke stažení škodlivého kódu až při spuštění instalace a už několikrát se mi tento předpoklad potvrdil, když jsem aplikaci z těchto zdrojů zkoušel nainstalovat a antivirový program při instalačním

procesu začal hlásit infikované soubory, které byly staženy právě samotným instalátorem až při jeho aktivaci. Podal jsem hlášení na podporu o distribuci závadného softwaru, ale reakce ze strany správců serveru nebyla žádná. I to částečně vypovídá o kvalitě těchto služeb a jejich úrovni zabezpečení, na kterou se rozhodně nelze spoléhat.

Vývojáři aplikací by se měli vyvarovat podobným technikám stahování součástí instalačních programů z internetu tzv. „za běhu“ a pokud je to nutné, je potřeba striktně dodržet maximální možnou úroveň zabezpečení takových přenosů. To znamená, že je potřeba ověřit certifikáty komunikujících stran a šifrovat kompletní probíhající komunikaci, aby nebylo možné ji nějakým způsobem narušit.

4.3 Opatrnost při instalaci

Instalační program obvykle vyžaduje zvýšená oprávnění pro přístup k systémovým zdrojům a k provádění nestandardních operací jako jsou zápisy do systémových adresářů, změny systémových souborů a nastavení apod. Než uživatel přidělí takovému instalačnímu programu zvýšené oprávnění, je vhodné zvážit možná rizika s přihlédnutím k původu softwaru a případně podrobit takový program testu nejprve na virtuálním stroji nebo aspoň v chráněném režimu, který poskytují některé bezpečnostní nástroje, které zajistí spuštění programu v odděleném prostředí a i přes vyžádání zvýšených oprávnění je takový program značně omezen v tom, co může se systémem provádět a v případě, že by mohlo dojít ke způsobení škod, uživatel má stále možnost zasáhnout a běh programu kontrolovat, případně odvrátit provedené změny. Problém je bohužel hlavně v samotných uživatelích, kteří dialogové okno oznamující takto závažné záležitosti jako je nedůvěryhodný zdroj aplikace, která není digitálně podepsána od důvěryhodného vydavatele, nebo požadavek na zvýšená oprávnění pro přístup k systému, ignorují a považují ho za pouhé pro ně samotné naprosto zbytečné zdržení, kterého je třeba se co nejdříve zbavit odsouhlasením všeho, co je ve zobrazeném upozornění napsáno. Proto je na místě hlavně snaha zvyšovat znalosti uživatelů a ukázat jim, proč se nevyplatí tyto bezpečnostní zásady podceňovat.

4.3.1 Deaktivace připojení k internetu

Při instalaci se může stát, že instalátor programu se pokusí přistupovat k internetovému připojení a bude se snažit komunikovat s vnějším světem. V takovém případě se může jednat o jakoukoliv komunikaci, která může buď zajišťovat odeslání citlivých dat o systému samotném na vzdálený a hlavně neznámý server nebo naopak může dojít k pokusu o stažení přídavných dat a programů, které mohou obsahovat škodlivý kód. V případě, že uživatel nemá správně nastavenou ochranu svého počítače s pomocí bezpečnostního software, je na místě aspoň po dobu běhu instalačního procesu deaktivovat připojení k internetu jeho zakázáním ve správci internetového připojení nebo přímo odpojením síťového kabelu z počítače resp. vypnutím adaptéru pro bezdrátové připojení.

4.3.2 Podezřelé nabídky přídavných programů

Součástí instalací u podobných volně dostupných programů bývají často nabídky dalšího software, který uživatel může nainstalovat společně s aktuálně instalovaným programem. Jedná se obvykle jen o snahu co nejvíce rozšířit nějaký program, který by si uživatel za normálních okolností vůbec

nepotřeboval do svého počítače nainstalovat, ale když je mu takto podstrčen spolu s jiným programem, situace se mění a vzhledem k tomu, že nastavení v průvodci instalací je obvykle vnímáno podobně jako dialogová okna s upozorněním, uživatelé ho naprosto ignorují, nezkoumají, co které nastavení umožňuje a jaké jsou aktuální hodnoty, které se potvrzují stiskem tlačítka Další. Pak se velice snadno stane situace, kdy uživatel nainstaluje do počítače spolu s chtěným programem několik dalších, které už nepožadoval, ale při instalaci „mlčením“ souhlasil s jejich přidáním do svého počítače. Důležitým podnětem pro ochranu uživatele je opět důraz jeho samotného na svou vlastní opatrnost a odpovědnost a především rozvážnost, která je v podobných situacích velice důležitá a těch několik sekund utracených čtením nastavení průvodce a obsahu dialogových oken může ve výsledku uživateli ušetřit mnohonásobně více času, než který touto rozvahou ztratí.

4.4 Používání kvalitního Firewallu

Firewall je software, který se primárně stará o zabezpečení, kontrolu a případné filtrování spojení počítače a na něm běžících programů s vnějším světem. Dobrý software tohoto druhu je schopen zabránit mnoha nebezpečným situacím, při kterých dochází ke škodlivé komunikaci, kdy se buď útočník snaží aktivně proniknout do systému z venčí s využitím nějaké zranitelnosti v samotném systému nebo některém z běžících programů, případně již nainstalovaný škodlivý software se snaží komunikovat se serverem útočníka, ze kterého případně může přijímat příkazy nebo stahovat další nebezpečný a škodlivý software. Precizní kontrolou spojení a jejich filtrací podle předem nastavených pravidel je možné tato rizika minimalizovat.[24]

Aby firewall dobře pracoval, je potřeba specifikovat právě výše zmíněná pravidla pro filtraci komunikace. Zde je opět kladen důraz na uživatelskou odpovědnost a znalost, protože právě po uživateli je obvykle požadováno, aby v případě zachycení pokusu o komunikaci některého z programů provedl rozhodnutí, zda takovou akci povolit či zakázat. Pravidla pro řízení komunikace lze nastavovat obvykle velice rozmanitě v závislosti na různých vstupních informacích, kterými můžou být síťové rozhraní, přes které se má komunikovat, protokol, který je ke komunikaci použitý, samozřejmě třeba adresa protistrany nebo použitý port a další i mnohem pokročilejší parametry. V případě chybného nastavení uživatel riskuje, že některé programy, které jsou z bezpečnostního hlediska v pořádku, nebudou správně pracovat, protože jim bude znemožněna potřebná komunikace po síti. Pokud se ovšem nastavení provede odborně s důrazem na prověření nainstalovaného software a striktního povolení pouze takové komunikace, která je skutečně potřeba, má takový krok velice příznivý vliv na zvýšení bezpečnosti celého systému

4.5 Použití antivirového software

Pro ochranu souborů a detekci škodlivého kódu, který se může do systému dostat při provozu ať už chybou uživatele nebo využitím existující zranitelnosti, slouží antivirový software, který by měl každý uživatel na svém stroji mít nainstalovaný se spuštěnou ochranou v reálném čase, která zajišťuje kontrolu procesů při jejich práci a je schopna často zabránit infiltraci škodlivého programu i na poslední chvíli. U antivirového programu je nejdůležitější udržovat ho stále aktuální, zejména jeho virovou databázi, neboť právě ta z něj činí účinný nástroj. Účinnost ale i přesto není nikdy stoprocentní, protože v případě výskytu nové nákazy, která ještě není uvedena v databázi antivirového programu, nejčastěji proto, že nový druh škodlivého kódu ještě nebyl analyzován vydavatelem

antivirového programu a tudíž takovou hrozbu nerozezná, nemusí být antivirový program účinný. Přítomnost antivirového programu v počítači neznamená, že nemůže být takový počítač ohrožen a úspěšně napaden. To by měl mít každý uživatel na paměti.

4.6 Opatření pro odstranění zranitelností

Při bezpečnostní analýze aplikace Kristýna byly zjištěny zranitelnosti, které umožňují útočníkovi získat přístup do napadeného zařízení. Vzniku podobných zranitelností lze efektivně předcházet dodržováním následujících zásad a protiopatření.

4.6.1 Opatření pro zabezpečení komunikace

Síťová komunikace programu, kam patří přesměrování na platební bránu a stahování instalčních balíčků, by měla být v každém případě realizována přes zabezpečený komunikační kanál, aby nebylo možné ze strany potenciálního útočníka komunikaci narušit. Zabezpečení komunikace lze realizovat následujícími kroky:

- Autentizace účastníků: obě strany by si měly navzájem ověřit svou identitu prostřednictvím certifikátu.
- Šifrování komunikace: výměna informací by měla probíhat s použitím dostatečně silného šifrování, které znemožní potenciálnímu útočníkovi rozluštit zachycené zprávy. V případě aplikace Kristýna by toto doporučení znamenalo komunikovat přes protokol HTTPS.

4.6.2 Opatření pro zabezpečení instalace a aktualizace

Samotný instalátor by v ideálním případě měl obsahovat všechny základní potřebné součásti aplikace, aby nebylo nutné připojovat se během instalačního procesu k síti a stahovat dodatečně přídavné součásti, protože tím vzniká riziko průniku škodlivého kódu do systému. Při aktualizaci se sice nevyhneme stažení nové verze programu, ale v tomto případě je důležité dbát na ověření stažených dat a předejít situaci, kdy se program pokusí spustit stažený soubor bez dostatečného ověření jeho pravosti.

4.6.3 Opatření pro zvýšení stability

Při návrhu a vývoji softwaru je potřeba se věnovat ošetření všech možných chybových stavů, které mohou při provozu nastat. Proto je důležité nejen software pečlivě navrhnout, ale také ho kvalitně testovat a to ideálně už během vývoje. Dodržení těchto zásad by mělo minimalizovat výskyt chyb, které způsobují nestabilitu programu a mohou vést dokonce ke vzniku zranitelností. V případě aplikace Kristýna by už při vývoji měla být zohledněna proměnlivá délka vstupních dat a pokud by k tomu nedošlo, měl by tento nedostatek být odhalen při testování.

Kapitola 5

Závěr

Úkolem v této práci bylo seznámit se s metodami a programovým vybavením používaným při zpětném inženýrství počítačového softwaru a následně provést bezpečnostní analýzu vybrané aplikace. Účelem bezpečnostní analýzy programu mělo být nalezení zranitelností a jejich následné zdokumentování a vyhodnocení. Následovat měl návrh vhodných protiopatření, která by vedla k nápravě zdokumentovaných zranitelností.

Úvodem byla nastíněna aktuální situace v oblasti bezpečnosti a vzhledem k předpokladu o budoucím vývoji je zde motivace zabývat se zabezpečením softwaru, které je v jistých aplikacích kritické. Byla stanovena kritéria výběru aplikace pro bezpečnostní analýzu, která musela být splněna. Jednalo se o oblíbenost mezi uživateli, původ programu a nedostupnost zveřejněných zdrojových kódů. Jméno aplikace, která splnila nejlépe zadaná kritéria, nemohlo být zveřejněno stejně jako další detaily, které by vedly k identifikaci zkoumané aplikace. Práce by v případě zveřejnění pravého jména aplikace mohla být použita jako návod k provedení útoku. Proto byly informace identifikující vybranou aplikaci odstraněny a jejím zástupným jménem je `Kristýna`.

V rámci příprav na bezpečnostní analýzu aplikace byly na základě požadavků na získávání informací o její činnosti vybrány vhodné programové nástroje. Pro monitorování síťového provozu se ukázal jako vhodný nástroj program Wireshark. Při analýze spuštěných procesů a prováděných operací byly využity nástroje z balíku Sysinternals Suite od společnosti Microsoft Corp. Konkrétně se jednalo o nástroje Process Explorer pro sledování spuštěných procesů, Process Monitor pro záznam činnosti procesů a nástroj Autoruns, který umožňoval sledovat a porovnávat výpisy oblastí sytémových registrů souvisejících se spouštěním úloh naplánovaných po startu operačního systému. Nástroj Dependency Walker se uplatnil při zjišťování problémů se závislostmi aplikace na dynamicky načítaných knihovnách a disassemblery s názvy OllyDBG, IDA Pro 5.0 a WinDBG umožnily analýzu kódu zkoumané aplikace a to jak statickou tak i dynamickou.

Provedení bezpečnostní analýzy vybrané aplikace zejména z hlediska zabezpečení síťové komunikace ukázalo, že aplikace i její instalační program obsahují poměrně závažné zranitelnosti, které mohou vést k úspěšnému napadení počítače, na kterém je aplikace instalována a provozována. Dle stanovených postupů, které vyplynuly z teoretické analýzy pro testování vybrané aplikace, bylo odhaleno několik zranitelností, které souvisely se špatným nebo vůbec žádným zabezpečením u přenášených dat ať už se jednalo o stahování spustitelných souborů určených pro instalaci doplňků či aktualizace programu nebo o komunikaci se vzdáleným serverem za účelem nákupu prémiového účtu s výhodami. Program projevil nestandardní chování při reakci na neočekávaně dlouhý vstupní

řetězec, které vzbudilo podezření na přítomnost další zranitelnosti, která by mohla umožňovat spustit útočníkovi škodlivý kód propašovaný na chybně nebo nedostatečně ošetřený vstup programu.

Všechny nalezené zranitelnosti byly zdokumentovány, popsány a došlo k jejich podrobnější analýze za účelem ověření možné zneužitelnosti pro případný útok. U většiny zranitelností se podezření potvrdilo a to zejména u zranitelností, které vyplývaly z nedostatečného zabezpečení komunikačních kanálů, přes které aplikace posílala a přijímala data od vzdáleného serveru za účelem aktualizace programu nebo provedení platby za prémiové členství. Při modelových útocích, které měly simulovat skutečné napadení klienta využívající zkoumaný software, bylo zjištěno, že je možné podvrhnout uživateli jakýkoliv spustitelný program, který bude po stažení aktivován a v případě instalátoru, který běží od svého startu v módu se zvýšeným oprávněním, dojde ke spuštění takto staženého škodlivého kódu rovněž se zvýšeným oprávněním. Stejně jednoduše je možné v případě pokusu o objednávku premiového účtu pro používání aplikace provést přesměrování klienta na škodlivou stránku, kde lze provést odcizení peněz určených na platbu objednané služby případně lze i při důvěřivosti a neopatrnosti uživatele odcizit přihlašovací údaje k platební bráně.

Pro nalezené zranitelnosti byla stanovena sada doporučení a bezpečnostních opatření, které by měly být dodržovány, aby rizika spojená s instalací a používáním nejen vybrané aplikace byla minimalizována nebo úplně odstraněna. Tato bezpečnostní opatření upozorňují zejména na nutnost odborného nastavení podpůrného software, který zajišťuje kontrolu probíhající komunikace a snaží se detekovat škodlivý kód, který by se mohl při provozu do systému dostat a zároveň dávají váhu i odpovědnosti uživatelů, po kterých je vyžadována opatrnost a rozvážnost při používání svých zařízení, která mohou být nesprávným používáním vystavena riziku úspěšného napadení útočníkem. Z velké části vzhledem k výsledkům práce a nalezeným zranitelnostem je odpovědnost i na samotných vývojářích softwaru, kteří by měli dbát na dodržování pravidel bezpečného programování a předcházet vzniku chyb, které ohrožují uživatele jejich programů.

Literatura

[1] *Statistika návštěvnosti* [online]. 2014. [cit. 8. 12. 2014]. Dostupné z: <http://www.netmonitor.cz>.

[2] *Status Code Definitions* [online]. 2014. [cit. 15. 4. 2015]. Dostupné z: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>.

[3] *Percentage of website traffic coming from mobile devices from 4th quarter 2010 to 1st quarter 2014* [online]. 2014. [cit. 26. 3. 2015]. Dostupné z: <http://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/>.

[4] *uniblue registry booster* [online]. 2010. [cit. 9. 1. 2015]. Dostupné z: <http://answers.microsoft.com/en-us/windows/forum/windows_vista-windows_programs/unibluereg istry-booster/f65b0a44-b501-4648-9625-3f7fc18ddde9>.

[5] *Is the program Uniblue Registry Booster a microsoft program?* [online]. 2012. [cit. 9. 1. 2015]. Dostupné z: <http://answers.microsoft.com/en-us/windows/forum/windows_7-windows_programs/is-theprogram-uniblue-registry-booster-a/977a4d07-c40e-4db9-bdc6-847756e7aa94>.

[6] *TZ Polovina všech uživatelů navštěvuje internet z mobilních zařízení* [online]. 2014. [cit. 28. 3. 2015]. Dostupné z: <http://www.netmonitor.cz/tz-polovina-vsech-uzivatelu-navstevuje-internet-z-mobilnich-zarizeni>.

[7] *What is PHP?* [online]. 2015. [cit. 4. 5. 2015]. Dostupné z: <http://php.net/manual/en/intro-whatis.php>.

[8] *Using Sysinternals tools like a pro* [online]. 2014. [cit. 31. 3. 2015]. Dostupné z: <http://www.howtogeek.com/school/sysinternals-pro/lesson2/>.

[9] *Statistika počtu stažení programů* [online]. 2014. [cit. 8. 12. 2014]. Dostupné z: <http://www.slunecnice.cz>.

[10] *Statistika počtu stažení programů* [online]. 2014. [cit. 8. 12. 2014]. Dostupné z: <http://www.stahuj.centrum.cz>.

[11] Chappell L. *Wireshark Network Analysis: The Official Wireshark Certified Network Analyst Study Guide*. Laura Chappell University, 2st edition, 2012. In English.

[12] Eilam E. *REVERSING Secrets of Reverse Engineering*. Wiley Publishing, Inc., 1st edition, 2005. In English.

[13] Howard M., LeBlanc D., Viega J. *24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them*. McGraw-Hill Osborne Media, 1st edition, 2009. In English.

[14] Kasper T. *Milking the Digital Cash Cow [29c3]* [online]. 2012. [cit. 2. 3. 2015]. Dostupné z: <https://www.youtube.com/watch?v=Y1o2ST03O8I>.

[15] Microsoft Corp. *Microsoft support policy for the use of registry cleaning utilities* [online]. 2014. [cit. 9. 1. 2015]. Dostupné z: <http://support.microsoft.com/kb/2563254/en-us>.

[16] Microsoft Corp. *Co je nástroj Obnovení systému?* [online]. 2015. [cit. 29. 3. 2015]. Dostupné z: <http://windows.microsoft.com/cs-cz/windows/what-is-system-restore#1TC=windows-7>.

[17] Microsoft Corp. *How Microsoft antimalware products identify malware and unwanted software* [online]. 2015. [cit. 4. 5. 2015]. Dostupné z: <http://www.microsoft.com/security/portal/mmpc/shared/objectivecriteria.aspx>.

[18] Microsoft Corp. *strcpy_s, wcscpy_s, _mbscpy_s* [online]. 2015. [cit. 20. 4. 2015]. Dostupné z: <https://msdn.microsoft.com/en-us/library/td1esda9.aspx>.

[19] Microsoft Corp. *Windows Sysinternals* [online]. 2015. [cit. 31. 3. 2015]. Dostupné z: <https://technet.microsoft.com/en-us/sysinternals/bb545021.aspx>.

[20] Peterka J. *Datové schránky používaly expirovaný certifikát, doplatily na to desítky tisíc zpráv* [online]. 2014. [cit. 2. 3. 2015]. Dostupné z: <http://www.lupa.cz/clanky/datove-schranky-pouzivaly-expirovany-certifikat-doplatily-na-to-desitky-tisic-zprav/>.

[21] Portnoy M. *Virtualization Essentials*. Sybex, 1st edition, 2012. In English.

[22] Rescorla E. *HTTP Over TLS* [online]. 2000. [cit. 5. 5. 2015]. Dostupné z: <http://tools.ietf.org/html/rfc2818>.

[23] Všetečka R. *Windows 8 ztratily podíl na trhu. Naopak rostly Windows 7 a XP* [online]. 2014. [cit. 28. 3. 2015]. Dostupné z: <http://technet.idnes.cz/windows-8-ztraci-na-ukor-windows-7-dqo-/software.aspx?c=A140703_171026_software_vse>.

[24] Walker M. *CEH Certified Ethical Hacker: Exam Guide (All-in-One)*. Mcgraw-Hill Osborne Media, 1st edition, 2011. In English.

[25] Zahradnický T. *Security and Secure programming 3. Buffer Overflow* [online]. 2013. [cit. 27. 3. 2015]. Dostupné z: <https://edux.fit.cvut.cz/courses/MI-BPR/_media/lectures/mi-bpr-2014-lecture-3.pdf>.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

ASLR Address Space Layout Randomization

CPU Central Processing Unit

CIL Common Intermediate Language

DDoS Distributed Denial of Service

DEP Data Execution Prevention

DoS Denial of Service

DLL Dynamic-Link Library

DNS Domain Name System

GPU Graphic Processing Unit

HTTP Hypertext Transfer Protocol

IP Internet Protocol

MD5 Message-Digest (algorithm) 5

OS Operation System

PHP Personal Home Page

PUP Potentially unwanted program

PUA Potentially unwanted application

RPC Remote Procedure Call

SSL Secure Socket Layer

SW Software

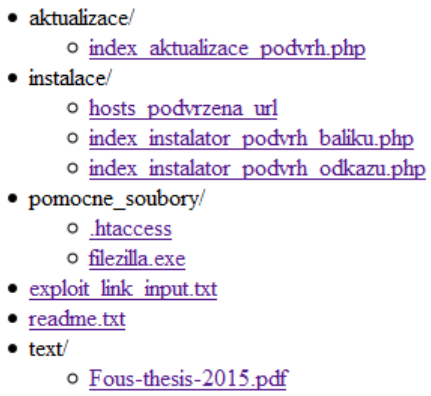
HW Hardware

⋮

Příloha B

Obsah přiloženého CD

Přiložené CD obsahuje zdrojové kódy pomocných podprogramů, skriptů a konfigurační soubory, které se nalézají ve složce `/instalace`, `/aktualizace` a `/pomocne_soubory`. Text této práce je ve formátu PDF k dispozici v adresáři `/text`. V kořenu CD je přítomen soubor index.html. Jedná se o statickou HTML stránku, která poskytuje navigaci pro procházení obsahu elektronické přílohy. Celý textový obsah je rovněž přítomen v souboru readme.txt, ve kterém je abstrakt a struktura adresářů na CD.



Obrázek B.1: Struktura přiloženého CD