Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Diplomová práce

Analýza kryptoviru

Bc. Jan Řečínský

Vedoucí práce: Ing. Josef Kokeš

21. prosince 2015

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) 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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 21. prosince 2015 .....

Odkaz na tuto práci

Řečínský, Jan. *Analýza kryptoviru.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato diplomová práce má za cíl analyzovat známý škodlivý kód CBT-Locker. Škodlivý kód je vysoce nebezpečný a byl velice medializovaný. Práce stručně uvede do principů Toru, Bitcoinu a nástrojů použitých při analýze. Následně se zabývá analýzou rodiny škodlivého kódu CBT-Locker a detailním rozborem jeho funkčního vzorku.

Klíčová slova analýza, bezpečnostní rizika, CBT-Locker, reverzní inženýrství, škodlivý kód

Abstract

The goal of this diploma thesis is to analyze the infamous malware CBT-Locker, which received significant attention of the media in the recent years. The thesis first introduces the principles of Tor, Bitcoin and the tools used to analyze the CBT-Locker code. Next, the thesis discusses in detail the examination of a functional CBT-Locker code.

Keywords analysis, CBT-Locker, malicious code, reverse engineering, security risks

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Snad každý uživatel počítače se setkal s nějakým druhem bezpečnostní hrozby. Především uživatelé operačního systému Microsoft Windows se mohou setkat s různými druhy virů, trojských koní, malwaru atd. V roce 2014 zasáhla Českou republiku vlna podvodných e-mailů, které odkazovaly na falešné stránky napodobující oficiální webové stránky České pošty. Falešné stránky obsahovaly odkaz na stažení škodlivého kódu. Informace o tomto malwaru proběhla velice intenzivně také médii jak odbornými [1], tak veřejnými [2] [3]. Většina běžných uživatelů tento e-mail otevřela, což je zřetelné z míry medialiace. Po spuštění malware pomocí silného šifrování znepřístupní to nejcennější, co uživatel má, tedy jeho data. Malware dorazil do několika e-mailových schránek lidí v mém okolí. Zaslechl jsem také o několika poškozených. Vzhledem k tomu, že malware zasáhl také mé okolí, rozhodl jsem se pro toto téma. Cílem této práce je analýza chování malwaru z podvodného e-mailu a zhodnocení bezpečnostích rizik, která malware CBT-Locker všeobecně představuje.

Kapitola 1

Projevy malwaru z pohledu oběti

Práce se zabývá analyzováním malwaru, který jen znám pod názvem CBT-Locker nebo Torrent Locker. Tato kapitola popisuje způsob, kterým malware infiltruje napadený systém. Dále popíše průvodní jevy a následky běhu malwaru.

1.1 Popis chování

Do povědomí se malware v České republice dostal šířením podvodných e-mailů[2] vydávajících se za informační e-mail[1] o nedoručeném balíku. Obsahoval odkaz na stránky vydávající se za oficiální webové stánky České pošty. Na těchto stránkách byl odkaz na stažení informací ohledně nedoručeného balíku. Po kliknutí na odkaz se stáhl soubor–archiv. Rozbalením archivu se zobrazil soubor, který vypadal jako PDF (viz obrázek 1.1). Pro běžného uživatele to nebylo nic podezřelého. Při bližším zkoumání nebyl tento soubor PDF souborem, ale spustitelným programem pro operační systém Windows. V domnění, že uživatelé otevřou PDF soubor s detailními informacemi o doručovaném balíku, spustili škodlivý kód.

Při kliknutí na tento soubor se program spustil. Po spuštění se pro obyčejného uživatele nic neděje. Vše se odehrává bez výraznějšího rušení jeho práce. Tu může narušovat pouze zvýšený hluk od běžícího ventilátoru. Pokud malware úspěšně ukončí svoji činnost, minimalizuje všechna otevřená okna a zobrazí vlastní okno s informacemi o zašifrování a možnostech zaplacení částky pomocí Bitcoinu. Zároveň také změní pozadí plochy. Zde trochu duplikuje okno, které je otevřené a zobrazuje zde téměř stejné informace ve zkráceném přehledu.

V okně je také běžící časomíra, která upozorňuje na lhůtu pro zaplacení. Tato lhůta vyjadřuje dobu, po kterou je možné zaplatit a obnovit zašifrovaná

Obrázek 1.1: Ikona spustitelného souboru vydávající se za PDF dokument

data. Toto odpočítávání času představuje velký psychologický nátlak na uživatele, který má pouze 96 hodin na to, aby se rozhodl, jak řešit vzniklý problém. Pokud nestihne převést požadovanou částku na danou Bitcoin adresu, již není možnost získat klíč a dešifrovat soubory. Pokud tedy autoři malwaru neposkytnou klíč i po uplynutí doby a opravdu ho smažou, data jsou nenávratně ztracena a jediná šance na obnovu souborů je z uživatelových záloh. Kompletní schéma útoku malwaru viz obrázek 1.2.

Rozbor malwaru jsem prováděl několika metodami reverzního inženýrství. Každá metoda je schopna odhalit různé vlastnosti zkoumaného programu. Musel jsem se seznámit s jednotlivými výše zmíněnými nástroji a metodami analyzování škodlivého kódu.

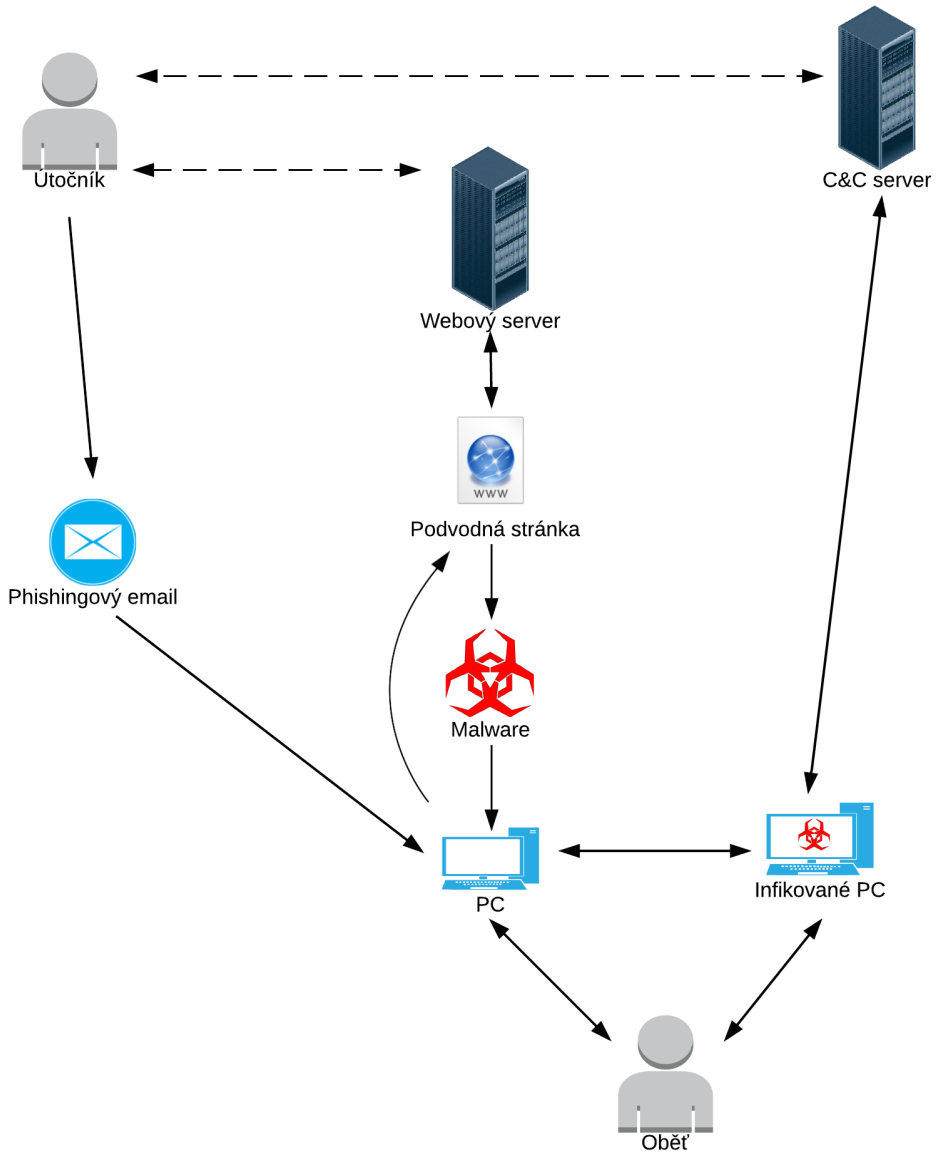
1.2 Vektor útoku malwaru CBT-Locker

V případě analyzovaných vzorků malwaru byl vždy vektor útoku[1] phishing, tedy zasílání podvodných e-mailů. Zde se mnoho verzí rozchází a využívají se různé techniky. Velice častá technika je pouhé rozesílání pomocí spamu. V tomto případě šlo konkrétně o phishing na Českou poštu.

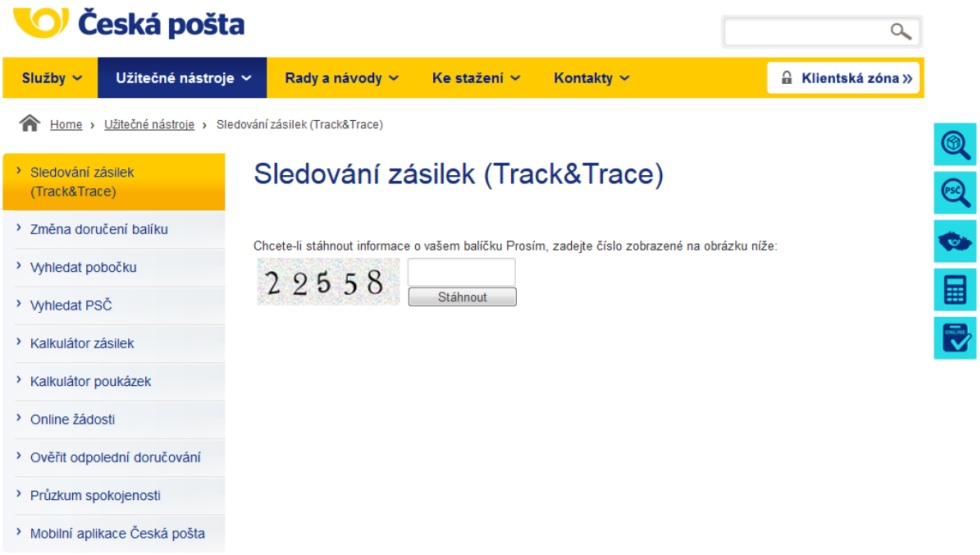
E-mail byl, bez gramatických chyb a na první pohled vystupoval velice důvěryhodně. Z tohoto hlediska se může vektor útoku klasifikovat jako velice zdařilý, profesionální a promyšlený. Následně byl uživatel přesměrován na podvodnou stránku (viz obrázek 1.3). Pokud tuto obrazovku porovnáme s originální stránkou České pošty (viz obrázek 1.4), je zde pouze rozdíl ve formuláři pro vložení čísla balíku. Jinak je podvodná stránka téměř identická s původní od České pošty.

Stránka směřovala na doménu **cs-posta24.org** skrze protokol HTTP. Oficiální stránky České pošty jsou umístěny na doméně **ceskaposta.cz** či **cpost.cz** . Pokud porovnáme originální a reálnou adresu, obě obsahují slovo „posta“ a v jisté podobě také „ceska“. Z velké podobnosti obou adres je

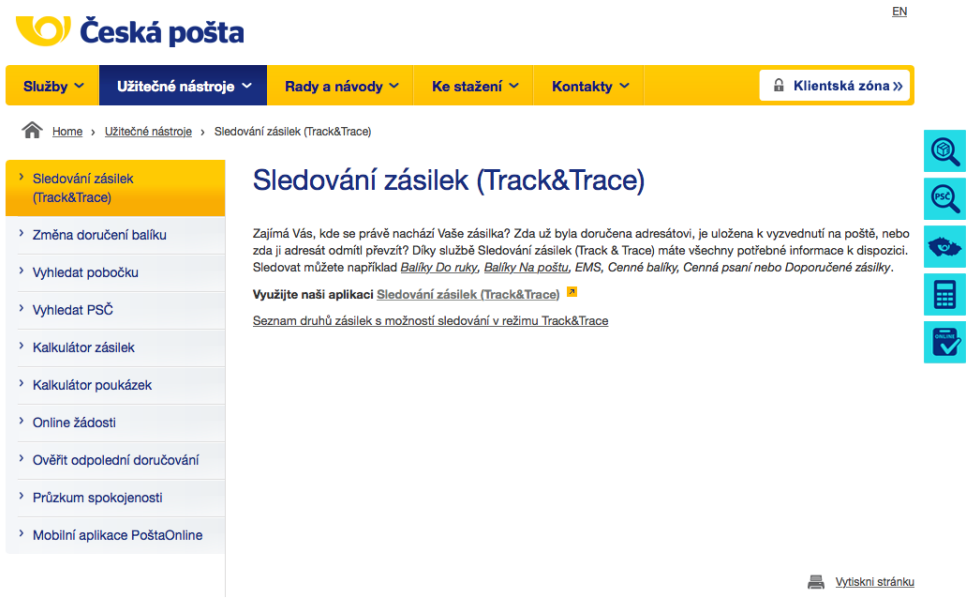
[1] Vektor počítačového útoku definuje způsob, jakým škodlivý kód pronikne do počítače, na který je útok cílen.



Obrázek 1.2: Diagram útoku malwaru CBT-Locker



Obrázek 1.3: Podvodná webová prezentace vydávající se za Českou poštu



Obrázek 1.4: Originální webová prezentace České pošty ze dne 6. 1. 2015[4]

zřejmá velká možnost zaměnitelnosti pravé adresy za falešnou, aniž by oběť pojala jakékoliv podezření.

1.3 Cíl útoku

Útok byl především cílen na méně zkušené uživatele počítačů, kteří nejsou seznámeni s riziky příloh a spouštění neznámých programů. Zde také bylo důležité, aby si oběť nevytvářela pravidelné zálohy. Tyto všechny faktory splňují již zmínění méně zkušení uživatelé. Malware ale cílil také na firemní e-maily. Firmy, které budou tímto malwarem napadeny, jsou pro útočníka mnohem cennější proto, že malware je schopen zasáhnout a ohrozit mnohem cennější data než u běžných uživatelů. Motivace firmy pro zaplacení je následně mnohem větší, a tím pádem má útočník mnohem vyšší šanci dosáhnout svého cíle, tedy získat finanční částku. Bohužel i ve firemním prostředí se často vyskytují nepoučení a nezkušení uživatelé, kteří zapříčiní znepřístupnění cenných dat.

Kapitola 2

Uvedení do problematiky Tor a Bitcoinu

2.1 Tor: The Onion Routing

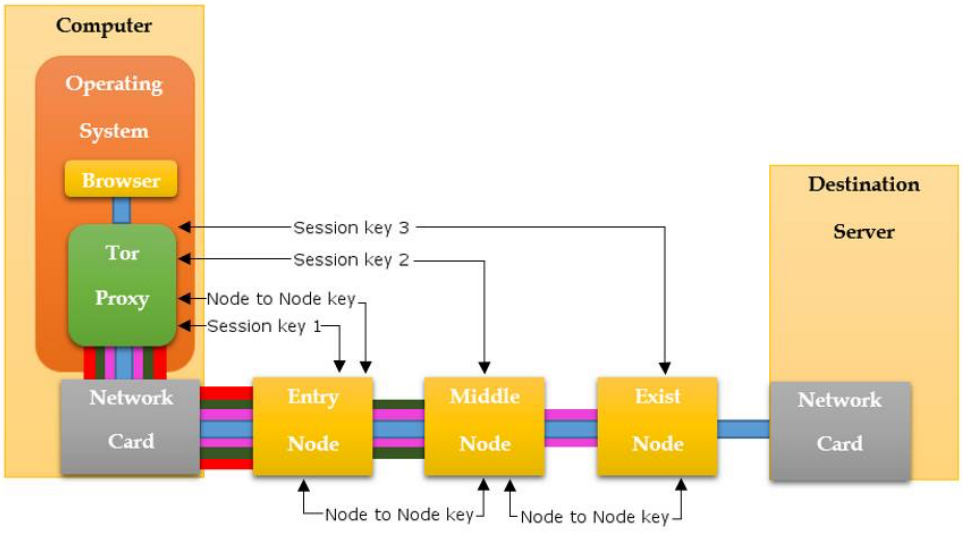
Jednou z chybějících vlastností internetu bylo, že neposkytoval dostatečný stupeň ochrany soukromí, tedy anonymity. Tuto chybějící vlastnost chtěli uživatelům internetu poskytnout autoři projektu Tor a skrze svůj projekt poskytnout absolutní anonymitu pro přístup ke službám přes internet. Tor umožňuje anonymně přistupovat k webovým službám. Anonymita zde může být využita jak k legitimnímu použití (zakrytí své indetity před sledovacími systémy, cílené reklamě...), tak i k nelegitimnímu (prodej drog, zbraní atd..). Toho jsou si autoři malwaru vědomi a využívají anonymizační síť Tor pro skrytí své pravé identity, k zakrytí pravé osoby, která stojí za vytvořením/řízením malwaru, ale také pro komunikaci s vytvořeným malwarem. Proto je vhodné zde čtenáře seznámit se základní funkčností sítě Tor.

2.1.1 Historie

Základ projektu Tor položili Paul Syverson, Michael G. Reed a David Goldschlag, zaměstnanci U.S. Naval Research Laboratory. Hlavním cílem bylo ochránit americkou online komunikaci. Dále byl Tor rozvíjen v roce 1997 společností DARPA, která projekt rozvinula v oblasti směrování v síti Onion routing. Alfa verze projektu, kterou známe nyní, byla vyvinuta v roce 2002 autory Rogerem Dingledinem a Nickem Mathewsonem Syversonem pod jménem The Onion Routing project, tedy ve zkratce Tor. Projekt Tor byl v současné podobě představen na konferenci 13th USENIX Security Symposium v roce 2004.

2.1.2 Popis funkčnosti

V předchozí kapitole bylo zmíněno, že hlavním cílem projektu Tor je poskytnout uživateli vysoký stupeň anonymity. Síť Tor se skládá z velkého počtu nodů. Tyto nody jsou klasické servery s nainstalovaným softwarem Tor. Nody rozdělujeme na tři druhy: vstupní, výstupní a tranzitní. Vstupní je první node v síti, se kterým klient komunikuje. Výstupní node komunikuje s cílovou službou místo klienta. Vysoká úroveň anonymity je získána pomocí několikanásobného přeposlání dat skrze jednotlivé tranzitní uzly sítě. Jednotlivé uzly znají pouze uzel, od kterého data obdržely, a uzel, kterému mají data poslat. Tedy pokud je uzel na cestě v síti na pozici N, zná pouze uzly N-1 a N+1. Cesta dat skrze síť Tor je volena zcela náhodně. Znovu vytvořená spojení přes síť Tor budou probíhat přes jiné náhodně zvolené nody. Mezi dvěma konkrétními uzly jsou data vždy znovu šifrována. Klíče od zašifrovaných dat jsou známy pouze dvěma vzájemně komunikujícím nodům, které prováděly příslušné šifrování, tzv. Node to Node key viz obrázek 2.1. Data jsou navíc šifrována přímo mezi klientem sítě Tor a jednotlivými nody pomocí tzv. Session key. Tímto několikavrstevným šifrováním je struktura přirovnávána k cibuli. Vrstvy šifrování je zcela jasně zřetelné na obrázku 2.1. Samotná data jsou mezi jednotlivými nody znázorněna modrou barvou. Následně jednotlivé vrstvy šifrování zobrazeny např. červenou, zelenou a růžovou barvou. Zde je již zřetelně viditelná zmíněná „cibulová“ struktura. Společně s náhodným přeposíláním umožňuje zaručit vysoký stupeň anonymity.



Obrázek 2.1: Schéma použití jednotlivých klíčů v síti Tor[5].

2.1.3 Zabezpečení anonymity

Nemůžeme zde spoléhat pouze na software Tor. Pokud použijeme nevhodný prohlížeč, můžeme celou anonymizaci znehodnotit. To může nastat, pokud webový prohlížeč, který je používán k běžnému prohlížení webu, použijeme v síti Tor. Následně se mohou informace např. o konfiguraci počítače, cookie, které poskytuje prohlížeč, použít pro odhalení indentity v síti Tor.

2.1.4 .onion adresy

Často se můžeme setkat s adresami s top-level doménou **.onion**. Např. `http://3fyb44wdhnd2ghhl.onion/wiki/index.php?title=Main_Page`. Tyto adresy nejsou dosažitelné přes kořen internetového DNS. Bez pomocného programu jsou nedostupné. Pomocí speciálního programu (např. Tor prohlížeče) jsou domény končící na **.onion** dosažitelné a také přes proxy servery, jako je např. Tor2web, které umožňují přistupovat do Tor sítě ze standardních internetových prohlížečů. Požadavek je poslán přes anonymizační síť Tor. Využití tohoto principu je opět především z důvodu anonymizace, tedy zastření identity poskytovatele služby, na kterou ukazuje .onion doména, ale také uživatele, který na tuto službu přistupuje. Adresy s doménou .onion jsou automaticky generovány Tor klienty. Společně s adresou je generován veřejný PGP klíč, který má délku 16 znaků. Název **.onion** je vztažen k Onion routování, které je využíváno v síti Tor.

2.1.5 Využití v malware

Tor využívají tvůrci malware především pro svoji ochranu, aby zamezili svojí identifikaci a případným právním postihům. Tuto síť používají především pro zakrytí komunikace mezi malwarem a C&C servery, ale také pro komunikaci s oběťmi malwaru.

2.2 Bitcoin

Bankovní účty, kreditní karty a další standardní finanční metody jsou vedeny na konkrétního člověka či firmu. Jednotlivé bankovní instituce si vedou záznamy o transakcích. V klasickém bankovním systému jsou obě strany transakce nuceny důvěřovat bankovní instituci(„třetí strana“). A proto nezávislá virtuální měna Bitcoin[6] zaujala volné místo mezi standardními finančními systémy – vypustila bankovní instituci a nahradila ji kryptografickými principy, které jsou všeobecně ověřitelné. Tím vylučuje jakoukoliv nucenou kontrolu či předání informací dalším stranám. Platba se provádí přes síť Bitcoinu přímo mezi odesílatelem a příjemcem zadané částky. Pro provedení transakce se používají ověřené kryptografické pricipy, které zaručují funkčnost a bezpečnost. Systém má architekturu peer-to-peer. Transakce se provádějí přímo

bez prostředníka. Pouze je nutné, aby transakce byla ověřena sítí a zaznamenána do sdílené databáze Block chain. Transakce může obsahovat více cílů, které obdrží definovanou částku. Jednotkou měny je Bitcoin. Bitcoin můžeme detailněji definovat jako první decentralizovanou virtuální měnu.

2.2.1 Historie

Bitcoin je online virtuální měna, která byla vyvinuta autorem[7], který vystupuje pod pseudonymem Satoshi Nakamoto. Totožnost člověka vystupujícího pod tímto pseudonymem nebyla dosud odhalena. K zveřejnění principů došlo v roce 2008. V lednu roku 2009 byl vytvořen první blok tzv. Genesis a byla provedena první platba v síti mezi Satoshi Nakamotem a Halem Finneym. Za průlom v používání měny Bitcoin se považuje událost z roku 2010, kdy byla pomocí Bitcoinu zaplacena pizza[7].

2.2.2 Block chain

Block chain je veřejná distribuovaná databáze informací o proběhlých transakcích. Částečnou kopii Block chainu obsahuje každý uzel sítě Bitcoin. Transakce obsahuje informace o tom, že uživatel **A** zaplatil částku **Y** uživateli **B**. Tato informace o transakci je distribuována do celé sítě. Databáze je sdílená mezi jednotlivými uzly sítě. Plná kopie Block chainu obsahuje každou transakci, která kdy byla uskutečněna v rámci měny. Každý blok obsahuje hash předcházejícího bloku a digitální podpisy vlastníků odesílaných částek. Toto má za následek vytvoření chronologického řetězce bloků od 1. bloku, tzv. Genesis, až po aktuální blok. Princip využití Block chainu je největší inovací, kterou virtuální měna Bitcoin přinesla. Díky němu nepotřebuje virtuální měna Bitcoin žádnou centrální autoritu. Samotný provoz měny provádí již zmíněná síť uzlů, na kterých běží Bitcoin software potvrzující jednotlivé platby.

2.2.3 Jednotka

Základní jednotka systému virtuální měny je Bitcoin. Od roku 2014 značený zkratkou jako XBT nebo BTC. Nejmenší možnou částku, na kterou můžeme Bitcoin rozdělit, je 0.00000001 BTC. Pro tuto nejmenší jednotku Bitcoinu se vžil název Satoshi podle jména zakladatele.

2.2.4 Princip anonymity

Základním problémem anonymity Bitcoinu je, že každá transakce je veřejně logována. Tato informace je každému přístupná v Block chainu. Může se zdát, že s anonymitou nemá nic společného, a čtenář si může klast otázku, jakou to má výhodu proti klasickým finančním systémům, které informace svých zákazníků nezveřejňují. Zde si můžeme všimnout propastného rozdílu, a to

toho, že v Block chainu není uvedena indentita uživatelů. Samotná adresa nic neříká o identitě uživatele.

Teoreticky pomocí analýzy transakcí můžeme anonymní adresu spárovat s reálným uživatelem a měna tímto způsobem přestává být anonymní. Toto oslabení anonymity Bitcoinu mohou způsobit uživatelé neopatrným zacházením s adresami peněženek. Pokud se uživatel zaregistruje např. na fóru a zde uvede svoji reálnou identitu, je možno pomocí analýzy Block chainu a procházení např. fóra spárovat reálného uživatele s adresou a následně tedy získat informaci o platbách.

2.2.5 Využití v malwaru

Tvůrci malwaru velice rádi využívají měnu Bitcoin z mnoha důvodů. Jedním z hlavních důvodů je snadná anonymizace převodu částek. Anonymizace je v tomto případě užití velice podstatná. Platební metodu, která by neposkytovala dostatečnou anonymitu, by tvůrci samozřejmě nevyužili, protože by je ohrožovala.

Kapitola 3

Nástroje reverzního inženýrství na operačním systému Microsoft Windows

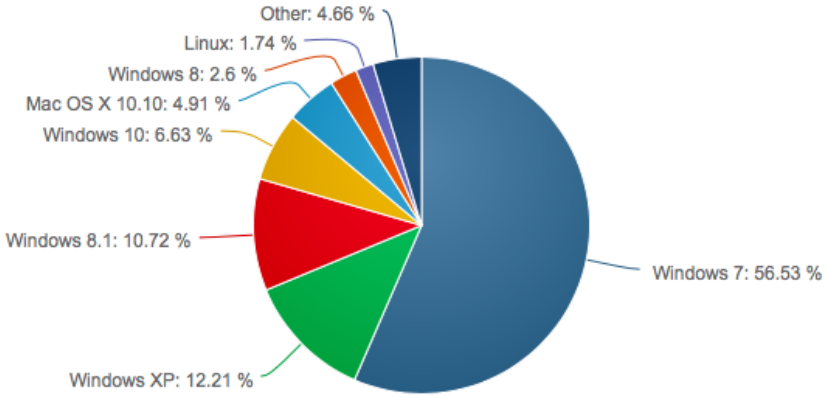
3.1 Microsoft vs. malware

Proč se velká část reverzního inženýrství aplikuje na operačním systému Microsoft Windows? Pokud si prohlédneme obrázek 3.1, je zde zřetelné, že nejvetší podíl na trhu s operačními systémy má společnost Microsoft se svými Windows. Vzhledem k takto masovému rozšíření není nijak překvapivé, že autoři škodlivého kódu míří z větší části na tento operační systém. Z tohoto důvodu je mnoho nástrojů vyvíjeno primárně pro operační systém MS Windows.

Analýza škodlivého kódu se provádí dvěma typy analýzy, a to statickou a dynamickou. Statická analýza se provádí na zkoumaném kódu, který neběží. Dynamická analýza se provádí narozdíl od statické analýzy na běžícím kódu. Tato metoda má několik výhod, ale i nevýhod proti statické analýze. Mezi hlavní úskalí dynamické analýzy patří, že analyzovaný program je spuštěn a nebrání mu téměř nic ve vykonávání svého kódu, který není znám. V případě, např. když se analyzuje malware, může program vykonat nebezpečné a v extrémním případě také nevratné změny v systému. Protikladem tohoto rizika je možnost detailního studia chování běžícího programu. Vzhledem k tomu, že se obě metody velice dobře doplňují, je vhodné obě metody kombinovat.

3.2 Prostředí

Pro reverzní inženýrství je velice podstatné zajistit si prostředí, které je dostatečně bezpečné a izolované. Takto uměle vytvořené prostředí umožňuje simulovat ideální podmínky pro běh škodlivého kódu. Je vhodné dostatečně



Obrázek 3.1: Rozšíření operačních systému pro osobní počítače[8].

izolovat analyzovaný kód především proto, že nevíme, jaké úmysly měl jeho autor.

3.2.1 Virtualizace

Virtualizace operačních systému umožňuje běžet OS v předem definovaných a uměle simulovaných podmínkách. Toto umožňují vytvořit např. programy VMWare a VirtualBox. Virtualizované prostředí poskytuje možnost vytvořit požadované prostředí, které je vyhovující pro reverzní analýzu kódu.

Virtualizační aplikace umožňují pořizovat tzv. snapshoty. Snapshoty slouží pro vytvoření zálohy před významnou událostí, jako je např. spuštění malwaru, zachycení zajímavé události či stavu virtualizovaného operačního systému. Díky této funkci si tyto události můžeme snadno uložit a později velice rychle obnovit a pokračovat v práci na zajímavé části reverzovaného kódu.

3.3 Použité nástroje

Pro obor reverzní inženýrství existuje velká škála nástrojů. Aplikace používané při reverzním inženýrství jsou velice nápomocné. Z velkého spektra nabízených programů jsem při analýze malwaru CBT-Locker využil níže detailně popsané aplikace. Pro každého, kdo se začne zabývat problematikou reverzního inženýrství, je mírný úvod do používaných nástrojů velice užitečný.

3.3.1 Olly dbg[2]

Olly dbg je freeware debugger pro 32-bitové aplikace určené pro operační systém Microsoft Windows. Olly dbg provádí analýzu nad binárním kódem. Umožňuje editaci binárního kódu za běhu zkoumané aplikace. Olly dbg zobrazuje jednotlivá volání funkcí Windows API, konstanty, stringy, tabulky, paměť atd. Olly dbg podporuje mnoho pluginů, které můžeme psát ve formě DLL souboru. Ohledně pluginů můžeme pozorovat největší rozdíl mezi verzemi Olly dbg 1.0 a 2.0. Komunita vytvořila mnoho pluginů pro verzi 1.0 a verze 2.0 se z tohoto hlediska nemůže poměřovat. Olly dbg se při analýze malwaru velice dobře osvědčil především přehledným zobrazením paměti, což bylo velice užitečné při detekci a popisu packeru. Naopak pro samotnou analýzu chování malwaru se program Olly dbg příliš neosvědčil, protože vzhledem k velikosti a komplexnosti kódu malwaru neposkytoval Olly dbg dostatečné nástroje pro pohodlnou práci jako např. IDA.

3.3.2 IDA[3]

IDA známá také jako Interactive Disassembler je program, který je hojně využíván v reverzním inženýrství. Program IDA je dostupný pro většinu používaných operačních systémů. IDA verze 5.0 je uvolněna pro nekomerční účely. Tato verze programu IDA byla použita v této práci. Od placené verze se liší především z hlediska chybějícího dekompilátoru Hex-Rays Decompiler. Dekompilátor je schopen jednotlivé analyzované funkce převést do pseudo-C podoby. Užitečným doplňkem k programu IDA je také kniha IDA Book[9], která se detailně zabývá seznámením s programem IDA. Já jsem z ní využil zvláště části jedna, dvě a tři, které mi pomohly při analýze malwaru. Pro samotné uvedení do problematiky reverzního inženýrství jsem využil knihu[10].

IDA provádí[11] analýzu strojového kódu metodou recursive traversal na rozdíl např. od WinDbg.

IDA na rozdíl od ostatních debuggerů/disassemblerů umožňuje zobrazení stavového diagramu programu. Diagram umožňuje názorně zobrazit strukturu funkce rozdělenou do několika tzv. basic blocks[4]. Tato funkce velice zjednodušuje orientaci v analyzovaném programu. Posloupnost a závislost jednolivých basic blocků je velice nápomocná při pochopení principů jednotlivých funkcí.

IDA také zobrazuje jednotlivé parametry známých API volání. Dokáže spojit jednotlivé sekce mezi sebou. Následně je schopna zobrazit diagram jednotlivých volání.

IDA nabízí rozšíření pomocí pluginů. Tyto pluginy jsou vyvíjeny komunitou okolo reverzního inženýrství a nabízejí nepřebernou možnost rozšíření funkčnosti programu IDA. Dále jsou uvedeny ty, které mě nejvíce zaujaly.

[2] `http://www.ollydbg.de`

[3] `https://www.hex-rays.com/products/ida/support/download_freeware.shtml`

[4] Posloupnost instrukcí vykonávaná bez jakéhokoliv přerušení

3.3.2.1 FindCrypt2[5]

Jedná se o plugin, který detekuje jednotlivé šifrovací a hashovací algoritmy. Detekce je prováděna tak, že většina šifrovaných algoritmů a hashovacích funkcí používá jedinečné konstanty, které jsou obsaženy v programu. Díky tomuto pluginu snadno odhalíme běžně používané algoritmy, jako jsou např: Camellia, CAST, CAST256, DES, GOST, RC6, Rijndael, SAFER, SHA-512 atd. Zároveň zobrazí adresu v programu, kde je konkrétně používán.

3.3.2.2 Fake code removal[6]

Tento plugin je velice užitečný, pokud autor programu použije obfuskaci[7] pomocí přebytečného kódu, který má za úkol zmást reverzního inženýra. Díky tomuto pluginu je odstraněn nepoužitý kód a zkoumaný kód se stává čitelnějším.

3.3.3 CFF Explorer[8]

CFF Explorer je program, který v sobě skrývá nepřeberné množství užitečných funkcí. Především se zaměřuje na přehledné zobrazení informací z PE hlavičky. Nepodporuje jenom zobrazování, ale také editaci. Při editaci PE hlaviček dohlíží na dodržení formátu a validity PE hlavičky(PE rebuilder), což je velice důležité, protože při nedodržení konzistence by upravovaný program již nebyl spustitelný. Kromě upravování PE nabízí ještě mnoho funkcí, jako jsou např: file scanner, memory dumper, process viewer atd. Program jsem především použil pro získání informací z PE hlavičky a její upravování.

Dále CFF Explorer umožňuje zobrazovat jednotlivé sekce a jejich vlastnosti, jako jsou např. název, velikost, začátek sekce, hash atd. U sekcí je velice zajímavé vyčíslit entropii obsahu sekcí. V oblasti malware je zajímavé pozorovat hodnotu entropie. Pokud sekce obsahuje zašifrovaná, či zapackerovaná data, entropie je větší než u plaintextu či spustitelného kódu, jak můžeme pozorovat v tabulce 3.1. Díky těmto získaným informacím lze velmi snadno identifikovat podezřelou sekci, na kterou je vhodné se později zaměřit. Ve většině případů v oblasti malwaru se jedná o nějaký druh packeru.

3.3.4 Sysinternals Utilities[9]

Sysinternals utilities je pokročilá skupina nástrojů, které slouží k hlubší analýze a diagnostice operačních systémů od společnosti Microsoft. Původně nebyl

[5] `http://www.hexblog.com/?p=28`

[6] `https://www.openrce.org/downloads/details/41/Fake_Code_Remover`

[7] Obfuskace je způsob upravení programu či strojového kódu takovým způsobem, který znesnadní jeho zpětnou analýzu a odhalení jeho chování nejenom nástroji pro to určenými, ale také pro samotným člověkem, který zkoumá program.

[8] `http://www.ntcore.com/exsuite.php`

[9] `https://technet.microsoft.com/en-us/sysinternals/bb545027.aspx`

Tabulka 3.1: Průměrné hodnoty entropie sekcí podle jejich obsahu. Hodnoty byly naměřeny z několika vzorků malwaru.

Druh	Entropie [b/B]
Text	4.347
Kód	5.099
Packer	6.801
Zašifrování	7.800

tento balík programů vyvíjen pod záštitou společnosti Microsoft, ale byl vytvořen v roce 1996 Markem Russinovichem. Z této sady užitečných programů byly při analýze malwaru CBT-Locker především využity nástroje Process Explorer, Process Monitor a Strings. Dále sada obsahuje např. PsTools, TcpView, BgInfo, BlueScreen.

3.3.4.1 Process Explorer

Process Explorer umožňuje podrobné sledování procesů běžících v operačním systému Microsoft Windows. Zobrazuje velice podrobný výpis procesů a jejich potomků. Process Explorer umožnil podrobné sledování procesů běžících v systému. Velice užitečné zde bylo zobrazení hierarchie procesů (jasně zřetelné zobrazení rodiče a jeho potomků). Tohoto bylo využito pro sledování základního chování malware a jeho chování v systému.

3.3.4.2 Process Monitor

Process Monitor je utilita, která zaznamenává aktivity procesu ohledně jeho chování v systému. Informuje o přístupu na souborový systém, registry či aktivitě procesu a jeho vláken. Událostí, které program Process Monitor zobrazuje, je mnoho, proto poskytuje uživatelsky příjemné filtry.

3.3.4.3 Strings

Program Strings umožňuje vyhledávat čitelné řetězce v programu. Pokud programy nevrátí žádné srozumitelné textové řetězce, je zde velká pravděpodobnost, že autor škodlivého kódu použil tzv. packer. Program se velice osvědčil při analýze jednolivých stupňů malwaru.

3.3.5 Regshot[10]

Regshot je velice užitečný program, pomocí něhož můžeme sledovat informace o změněných (přidaných/odebraných) údajích v položkách v registrech. Zde

[10] `https://code.google.com/p/regshot/`

volíme první tzv. shot, tedy okamžik, kdy je zaznamenán stav registrů. Následně zde zvolíme druhý shot. Potom nám program zobrazí velice přehledný seznam informací o registrech.

Funkci můžeme použít nejenom na systémové registry, ale také na souborový systém. Toto nám umožňuje detailně si zobrazit informace o modifikovaných údajích obsažených v registrech či modifikovaných souborech. Velmi užitečné využití má tato aplikace při sledování změn, které se udály při spouštění analyzovaného programu.

3.3.6 PE Explorer[11]

Následně u malwaru můžeme použít nástroj PE Explorer. Tento program nám zobrazí informace obsažené v PE hlavičce, např. datum sestavení, info o překladači, importech, exportech, metadatech, vstupním bodu atd. Tyto údaje poskytují mnoho informací ohledně programu, který je analyzován. Následně je program PE Explorer schopen detekovat ve spustitelném kódu různé druhy obfuskací, jako jsou např. packery. Díky této informaci může být obfuskace snadněji zdolána.

3.3.7 VirusTotal[12]

VirusTotal[12] je online analyzátor poskytující analýzu škodlivých kódů a podezřelých URL. Po nahrání kódu zobrazí konkrétní jméno, které je mu přiřazeno, a dodatečné informace, které jsou analýzou zjištěny. Tato služba může být využívána nejenom pro ověřování neznámých programů, ale také pro reverzní inženýry, kteří chtějí konkrétně identifikovat škodlivý kód, který zrovna zkoumají a již byl identifikován antivirovými firmami. Pokud jde o neidenfikovaný malware, VirusTotal není schopen poskytnou relevatní informace. VirusTotal získává informace o analyzovaném programu několika způsoby: antivirovými programy, webovými skenery, souborovou a URL analýzou a příspěvky od uživatelů. Tyto informace jsou např. hash souboru, základní chování, modifikované soubory a registry, entropie jednotlivých programových sekcí, IP adresy, se kterými komunikuje atd. Velice jednoduše získané informace nabývají na důležitosti právě proto, že VirusTotal shromažďuje tyto informace na jednom místě. Následně je velice snadné identifikovat různé rodiny škodlivého kódu, případně jednotlivé kampaně. VirusTotal dále zobrazuje seznam aktuálně používaných antivirových programů. V tomto seznamu je uvedeno jméno malwaru, kterým antivirový program identifikuje konkrétní druh/modifikaci škodlivého kódu a pod kterým je veden v databázi antivirové společnosti. Toto jméno je v seznamu uvedeno, pouze pokud je analyzovaný vzorek detekován jako škodlivý kód.

[11] `http://www.heaventools.com`
[12] `https://www.virustotal.com`

3.3.8 Charles[13]

Charles je HTTP proxy a monitor, který umožňuje pohodlně ladit HTTP komunikaci. Vzhledem k tomu, že Charles proxy poskytuje také možnost SSL proxy, je velice pohodlné tento nástroj používat pro její analýzu. Následně umožňuje zobrazovat časový diagram jednotlivých odeslaných a přijmutých HTTP požadavků. Přehledně zobrazuje HTTP hlavičky, které jsou velice podstatnou částí HTTP protokolu. Při použití SSL proxy také nabízí stáhnutí SSL certifikátu pro eliminaci hlášení o nedůvěryhodném certifikátu. Charles proxy se osvědčila pro odstranění šifrování z protokolu HTTPS. Toto odstranění by např. pomocí programu Wireshark bylo o poznání náročnější. Zároveň Charles poskytuje přehledně na jednom místě jednotlivé HTTP metody seřazeny pod jednotlivé domény. Takto přehledný seznam sniffer Wireshark neumožňuje zobrazit.

[13] `https://www.charlesproxy.com`

Kapitola 4

Ochrana kódu před reverzním inženýrstvím

Základním účelem ochrany kódu je skrýt kód před nežádoucím analyzováním, které má za cíl např. prolomení DRM ochrany, obcházení zakoupení softwaru či neautorizované kopírování, tedy úkony, se kterými autor softwaru nesouhlasí či jsou nelegální. Metod na ochranu softwaru je velké množství a zde bude pouze zmíněna malá část, která čtenáře uvede do problematiky obfuskací a zmíní specifické obfuskace používané u malwaru.

Ochrany můžeme členit dle [13] na tyto části:

- Právní ochrana
- Technická ochrana
  - Obfuskace
  - Šifrování
  - (Částečný) běh na serveru
  - Důvěryhodný nativní kód

Transformaci kódu můžeme členit na tyto kategorie:

- Obfuskace dat „data obfuscation“
- Obfuskace běhu programu „control obfuscation“
- Obfuskace struktury „layout obfuscation“
- Ztížení deobfuskovacích metod „preventive transformation“

4.1 Příklady obfuskací, obfuskace běhu programu

Do této sekce spadá především vkládání neaktivního kódu – „dead code“ nebo irelevatního kódu. Tento způsob obfuskace má za cíl zmást člověka, který analyzuje obfuskovaný kód.

Dále sem můžeme zařadit upravení podmínek cyklů – „extend loop conditions“. Toto upravení přinese do analýzy mnoho nadbytečného kódu, kde bude o poznání složitější určit přesný princip interování cyklu, a proto se velice zvýší obtížnost reverzní analýzy.

V neposlední řadě do této kategorie obfuskací patří: rozbalení cyklů, inlinování metod, klonování metod, přidání nadbytečných operandů, paralelizace.

Obfuskaci můžeme vidět na příkladu:

```
i=1;
while (i<100){
        ......
}
```

```
i=1;
j=100;
while ((i<100) && (j*j*(j+1)*(j+1)%4 == 0)) {
                  .....
                  i++;
                  j=j*i+3;
}
```

Převzato z [13]

4.2 Obfuskace specifické pro malware

Malware je velice specifický z hlediska obfuskací, a to z důvodu, že autoři malwaru se pokoušejí používat nejnovější a neobvyklé metody obfuskací proto, aby unikli detekci antivirů a malware byl ve svém působení co možná neúčinnější.

4.2.1 Šifrování

První malware využívající šifrování se objevil v roce 1986[14]. Základní obfuskace šifrováním může být realizována pomocí jednoduchých manipulací s bity, ale také náročnější pomocí známých kryptografických algoritmů, jako je např: DES, AES. Tato metoda obfuskace je velice oblíbená mezi tvůrci malwaru a jejím nasazováním se tvůrci malwarů především brání detekci antivirů.

4.2.2 XOR

Obfuskace pomocí binárního operátoru XOR je v malware velice oblíbená. Z důvodu své jednoduchosti a velice dobrého účinku na úroveň obfuskace se

také nazývá binární obfuskace, což pomáhá předcházení detekce antivirových programů. Jako alternativa k operátoru XOR se také používá kódování Base 64.

4.2.3 Runtime packers

Původním účelem packers bylo zmenšit velikost programu a místa, které zabíral. Tvůrci malwaru tuto metodu nevyužívají jen z důvodu změny své signatury a bránění antivirům, ale také jako ochranu před reverzním inženýrstvím. Všeobecně je známo několik packerů, jako je např PECompact, NeoLite, EXPressor. Tyto packery jsou volně přístupné. Tvůrci tyto známé packery nepoužívají a vytvářejí vlastní uzavřené. Zmínku, že studovaný program je obfuskován pomocí packeru, můžeme vysledovat z mnoha vedlejších efektů, které packer způsobí.

- **Stringy** Výstup neposkytne žádné relevatní výsledky.
- **Importy** Při analýze importů programu se vyskytuje pouze velice málo importů programu, a to především Kernel32.dll s metodami **LoadLibrary**, **GetProcAddress**,**VirtualAlloc** a **VirtualFree**.
- **Sekce** Zde je často velice nápadný jejich počet a velikost.
- **Entropie sekcí** Pokud spočítáme entropii jednotlivých sekcí, často je entropie větší než u běžného programu. Sekce bývá označena jako **executable**.

4.2.4 Anti-virtual Machine

Při studování malwaru se chce reverzní inženýr většinou vyhnout běhu malwaru na primárním operačním systému. Proto se zde využívají programy jako VirtualBox či VMware. Tvůrci malwaru jsou si vědomi, že studium jejich programů se provádí v těchto virtualizovaných prostředích, a z tohoto důvodu umisťují do kódu tyto ochrany. Především používají dvě metody: Tato metoda je založena na obsahu registru **TSC** (Time Stamp Counter). Tento registr obsahuje počet cyklů, které procesor vykonal od posledního restartu.

Pokud tato detekce běží na nevirtualizovaném operačním systému, nebývá rozdíl mezi instrukcemi větší než 100 ms. Pokud je operační systém virtualizovaný, je rozdíl mnohonásobně větší. Díky této detekci může malware přizpůsobit své chování.

Pokud je operační systém virtualizován např. v programu VMWare, v operačním systému Windows zanechá v registru

`HKLM\SYSTEMCurrentControlSet\Services\Disk\Enum`

hodnotu. Obvyklé je, že tuto hodnotu obsahují pouze systémy, které jsou virtualizované. Dále mohou virtulizované operační systémy obsahovat nástroje

pro podporu virtualizace, jako jsou např. **VMTools** nebo **VBoxVmService**. Malware může v systému detekovat tyto nástroje a upravit své chování.

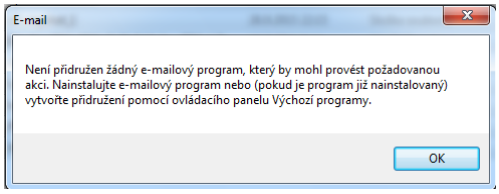
Kapitola 5

Analýza 1. vzorku

Tento vzorek byl získán z reálného případu napadení malwarem CBT-Locker, při kterém došlo k zašifrování velice podstatných nezazálohovaných dat. V tomto případě se jednalo o kampaň realizovanou skrze phishing České pošty, která byla uskutečněna na podzim roku 2014. Při incidentu došlo k zaplacení požadované částky prostřednictvím měny Bitcoin.

5.1 Ověření funkčnosti

Před samotným započetím analýzy bylo nutné ověřit funkčnost malwaru. Funkčnost byla ověřena na čistě instalovaném operačním systému Microsoft Windows 7. Na systému nebyl instalován žádný dodatečný software, zejména např. antivirus. Na testovaný operační systém byl přenesen 1. vzorek malwaru a spuštěn. Po spuštění analyzovaného malwaru se zobrazilo okno (viz obr.5.1) upozorňující na nepřítomnost mailového klienta. Po instalaci klienta se již okno nezobrazilo. Nyní se mohl celý test opakovat, pouze operační systém obsahoval monitorovací nástroje. Malware byl monitorován nástroji Process Monitor, Process Explorer, Wireshark a Charles proxy. Data získáná monitorováním jsou přiložena na CD. DNS dotaz odchycený programem Wireshark v souboru dns.pcapng, výpis činností malwaru monitorovaného pomocí Process Monitoru je uložen v souboru mon1.pml, záznam z monitorování provozu pomocí Charles proxy je zaznamenán obrázky v textu. Bohužel malware neprovedl žádné zašifrování souborů a ani nijak neinformoval o zašifrování. I přes zjevnou nefunkčnost bylo vhodné vyhodnotit nasbíraná data a provést základní analýzu malwaru.



Obrázek 5.1: Informace o nutnosti instalace e-mailového klienta

Address	Length	Type	String
.rdata:0041633C	00000005	C	!'\t>
.rdata:0041639C	00000005	C	|#<>
.rdata:004163CB	00000006	C	MVx:>
.rdata:0041672C	00000005	C	Jg >
.rdata:00416774	00000005	C	hb\">
.rdata:004167D4	00000005	C	\r\n@>
.rdata:0041681B	00000006	C	[j&,>
.rdata:0041687C	00000005	C	\bp&>
.rdata:0041697F	00000005	C	?\bR`D
.rdata:004169E4	00000005	C	2G<>
.rdata:00416ACF	0000000A	C	?6FID\x1B?9>
.rdata:00416AEC	00000005	C	gsL>
.rdata:00416B1C	00000005	C	09C>
.rdata:00416B4C	00000005	C	V\tC>
.rdata:00416BF4	00000005	C	NcB>
.rdata:00416DA3	00000006	C	:&\bJ>
.rdata:00416DB7	00000005	C	?5L$.
.rdata:00416EAC	00000005	C	+oF>
.rdata:00416F39	00000005	C	\aj\\&\"
.rdata:00416F9B	00000006	C	F\\IE>
.rdata:00417117	00000005	C	?A%My
.rdata:0041717C	00000005	C	f7O>
.rdata:004171C3	00000006	C	B'=>>
.rdata:0041721F	00000005	C	?QNT\t
.rdata:00417284	00000005	C	;„*>

Obrázek 5.2: Textové řetezce obsažené v 1. vzorku malwaru

5.2 Statická analýza

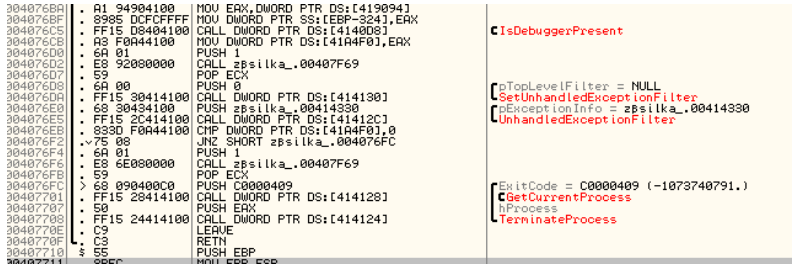
5.2.1 Strings

Pro tuto analýzu byl využit program IDA. Na obrázku 5.2 je viditelné, že výpis neobsahuje žádné smysluplné textové řetězce, které jsou zde očekávány, tedy řetězce, které by vypovídaly o chování programu. Chybějící smysluplné textové řetězce jsou znakem, že malware obsahuje jistou formu obfuskace trasformací, jako jsou např. XOR, Base64, šifrování či packer.

5.2.2 Importy knihoven

Výpis použitých knihoven jsme získali pomocí CFF exploreru, nebo je možné ho získat pomocí programu IDA.

- ADVAPI32.dll – přístup k registrům, restart systému atd.



Obrázek 5.3: Obfuskace pomocí isDebuggerPresent

- COMDLG32.dll – dialogové boxy, např. otevíraní souborů, tisk, vybíraní barev
- GDI32.dll – základní grafická knihovna
- KERNEL32.dll – základní operace s WIN32 API
- USER32.dll – správa oken (vytváření, manipulace), zobrazování oken

5.2.3 Obfuskace

Pouze základní analýzou škodlivého kódu je možné pozorovat několik základních obfuskací.

5.2.4 IsDebuggerPresent

Tato obfuskace byla zcela zřejmá již při samotném zahájení analýzy. Pokud se malware spustil v debuggeru bez jakéhokoliv breakpointu, skončil velice rychle a nezanechával v systému žádné změny např. v registrech či na souborovém systému. Následně při analýze v programu IDA bylo možno velice brzy pozorovat tuto základní obfuskaci (viz obrázek 5.3). Zmíněnou obfuskaci můžeme nalezt na adrese **0x4076C5**. Po otestování výsledku je zavolán blok pro ukončení procesu (viz obrázek 5.3).

5.2.5 PEheader

Při podrobnějším zkoumání vzorku pomocí CFF Explorer jsme si mohli všimnout několika velice zajímavých údajů, jako je časová značka.

- **TimeStamp** 2014/11/05 20:44:47 UTC
- **CompanyName** Adobe Systems Incorporated
- **FileDescription** Adobe Photo Downloader
- **FileVersion** 6.0.0.131330

- **InternalName** Adobe Photo Downloader

Pomocí popisu souboru je zcela zřetelné, že se malware pokouší v systému vydávat za software od firmy **Adobe**, a to konkrétně za **Photo Downloader**, a tímto „maskováním“ se snaží alespoň částečně zmást antivirové programy.

5.2.6 Programové sekce

Program obsahuje tyto sekce (viz tabulka 5.1). U sekce **.text** a **.rsrc** je možno pozorovat zvýšenou etropii, ale nepřekračuje mez, která by indikovala některý druh obfuskace.

Tabulka 5.1: Sekce a jejich entropie 1. vzorku malwaru

Název sekce	Entropie [b/B]
.text	6.34
.rdata	4.51
.rsrc	6.23

5.2.7 VirusTotal

Zde bylo vhodné využít pro získání všeobecných informací ohledně prvního zachycení čas první analýzy tohoto malwaru na server VirusTotal **First submission 2014-11-13 10:46:51 UTC**.

Jména, pod kterými se malware vyskytuje v databázi nástroje VirusTotal, uvádí následující přehled:

- uqygurap.exe
- PhotoDownloader.exe
- zásilka_32937298291.exe
- Adobe Photo Downloader.exe
- DHL_Versandschein.exe

Z uvedených názvů je patrné, že malware se pokoušel infiltrovat do počítačů jako soubor informující o zásilce od společnosti **DHL** či jako software od **Adobe Photo downloader**. Také zde vidíme český název **zásilka_32937298291.exe**. Pokud je datum zjištěné z PE hlavičky CFExplorer pravdivé, uplyne mezi sestavením malwaru a jeho prvním nahlášením na portál VirusTotal zhruba 8 dní.

Dále byla pomocí VirusTotal ověřena doména walkingdead32.ru, zda není klasifikována jako závadná. Ke dni 5. 11. 2015 byla klasifikována již pouze jedním antivirem, a to Fortinetem. Ostatních 64 antivirů klasifikovalo doménu

jako bezpečnou. Tento překvapivý neúspěch v klasifikování jako malware můžeme připsat nejenom nedokonalosti nástroje VirusTotal, ale také zejména stáří analyzovaného vzorku.

5.2.8 Změny v souborovém systému

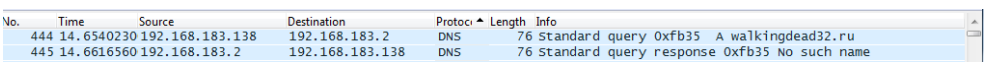
Jedinou změnou v souborovém systému napadeného počítače bylo vytvoření uvedeného souboru. `C:\Windows\efisubec.exe`.

5.2.9 Změny v registrech

`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\efisubec: C:\Windows\efisubec.exe` zde je možno si všimnout především zápisu do registru. Tento registr zajišťuje spuštění programu při každém přihlášení uživatele. Program, který se má spustit, je uveden v hodnotě registru. Cesta, která je zde uvedena, je také velice zajímavá, a to především proto, že uvedený soubor byl vytvořen spuštěným malwarem. Tímto zápisem do registru si malware vytváří podmínky pro trvalé přežití v systému. Pokud by tento úkon neučinil, zůstal by na místě, kam ho uživatel stáhl do systému, a mohlo by se velice snadno stát, že bude uživatelem odstraněn. Z toho vyplývají následky nemožnosti neúspěšného běhu. Bylo velice snadno ověřitelné, že malware provedl úkon zkopírování sám sebe do uvedené složky. Když se zobrazí informace o změnách na souborovém systému, je zcela zřetelné, že malware tady vytvořil soubor, ke kterému v registru odkazuje.

5.3 Komunikace po síti

Při analýze síťového provozu byly využity programy Wireshark a Charles proxy. Při analýze v programu Wireshark byla nalezena komunikace pomocí DNS protokolu, která se pokoušela přeložit doménové jméno **walkingdead32.ru** na IP adresu. Záznam této komunikace je uložen v souboru dns.pcapng. Ze záznamu bylo patrné (viz obrázek 5.4), že se překlad nepovedl, a doména tedy nebyla přeložena na IP adresu. Lze předpokládat, že IP adresa již není uvedena v DNS záznamu. Pro ověření neexistence IP adresy v DNS záznamu byl využit konzolový příkaz dig. Tento příkaz přeloží doménu na IP adresu, která je obsažena v DNS záznamu. Výsledek programu dig nevrátil žádné záznamy, což znamenalo, že jsme si potvrdili předpoklad neexistence IP adresy v záznamu. V důsledku toho není malware schopen komunikovat s C&C severem, a může to být jeden z důvodů, proč není funkční.



Obrázek 5.4: Neúspěšný DNS dotaz

Dále bylo vhodné použít další doménový nástroj Whois. Pomocí tohoto nástroje byla získána informace ohledně vlastníka domény a registrátora, u kterého byla registrace provedena.

```
domain:          WALKINGDEAD32.RU
nserver:         ns1.reg.ru.
nserver:         ns2.reg.ru.
state:           REGISTERED, NOT DELEGATED, UNVERIFIED
person:          Private Person
registrar:       REGRU-RU
admin-contact:   http://www.reg.ru/whois/admin\_contact
created:         2014.11.07
paid-till:       2015.11.07
free-date:       2015.12.08
source:          TCI
```

Z výpisu můžeme vyčíst, že doména byla zaregistrována dne 7. 11. 2014 u ruského registrátora **reg.ru**. Bohužel, informace ohledně osoby, která stála za zřízením zmíněné domény, nebylo možné detailněji dohledat a osoba nebyla odhalena.

Nadále bylo vhodnější použít Charles proxy z důvodu, že malware používá pro komunikaci šifrovaný protokol HTTP. Charles proxy nabízí tzv. SSL proxy pro HTTPS protokol. Pro komunikaci se serverem využívá tedy vlastní SSL certifikát, a proto umožňuje zobrazení dešifrované komunikace. Nyní pomocí Charles proxy se zapnutým doplňkem SSL proxy bylo možno pozorovat, že komunikace s výše uvedenou doménou není úspěšná a HTTP metoda POST vrací chybu na vypršení spojení.

5.3.1 Vytvoření vlastního C&C serveru

Z minulé sekce je zřejmé, že C&C server je již nedostupný. Z tohoto důvodu byl sám analyzovaný malware také nefunkční. Proto zde bylo vhodné pokusit se o vytvoření vlastního C&C serveru.

Pomocí předchozí analýzy komunikace bylo zřejmé, že malware se pokouší komunikovat s doménou **walkingdead32.ru**. Prvním úkolem bylo zajistit překlad tohoto doménového jména na lokální experimentální server. Toto bylo realizování skrze DNS server běžící na routeru. Zde byl jako IP adresa zadán lokální testovací server. Při této úpravě a opětovné analýze komunikace bylo zřejmé, že bude nutné implementovat HTTP server podporující šifrování. K tomuto účelu byl využit webserver Apache2 a self-signed certifikáty. Při opětovném prozkoumání provozu Charles proxy vykazovala opět nepodporující certifikát. Tento problém se již týkal pouze programu Cherles proxy a webového serveru. Tento problém bylo nutné vyřešit záměnou webového serveru za webový server NGINX. NGINX tuto chybu neobsahoval a spolehlivě spolupracoval s programem Charles proxy. Nyní už bylo zřejmé, že malware se

pokouší komunikovat na URL **https://walkingdead32.ru/topic.php** pomocí HTTPS protokolu metodou POST. Nyní bylo potřeba zajistit na webovém serveru, aby taková URL existovala. Toto bylo splněno velice jednoduše, a to pomocí PHP skriptu s názvem topic.php umístěného do kořene webového serveru. Následně analýza komunikace vykazovala úspěch, a to díky HTTP návratovému kódu 200 (viz obrázek 5.5), tedy vše proběhlo v pořádku. Z obrázku 5.6 je zřetelné, že malware zasílá hostname napadeného počítače a další neidentifikované informace. Po takto jednoduše napsaném vlastním C&C serveru bohužel malware opět nezahájil svoji činnost. Toto mohlo mít mnoho příčin. Jedním z důvodů mohl být nedokonale napsaný vlastní C&C server. Vytvořený C&C nevracel v HTTP metodě požadovaná data. Formát požadovaných dat by se mohl získat deailní reverzní analýzou malwaru. Vzhledem k jeho základní nefunkčnosti by analýza nemusela přinést požadované výsledky a její provádění by bylo zbytečné.

Server tedy nevracel v HTTP metodě požadovaná data. Dále mohl mít malware v sobě zakódovanou dobu, kdy má být aktivní a může napadat systém. Zakódovaná doba funkčnosti mohla být závislá na systémovém času. Proto zde bylo vhodné vyzkoušet změnu systémového času. Bohužel, tato změna neměla na funkčnost malwaru žádný vliv a malware zůstal neaktinvní.

Name	Value
URL	https://walkingdead32.ru/topic.php
Status	Complete
Response Code	200 OK
Protocol	HTTP/1.1
Method	POST
Kept Alive	No
Content-Type	text/html; charset=UTF-8
Client Address	/127.0.0.1
Remote Address	walkingdead32.ru/192.168.88.8

Obrázek 5.5: Uspěšné navázání komunikace s vlastním C&C serverem

```
00000000  57 00 49 00 4e 00 2d 00 56 00 54 00 39 00 49 00   W I N - V T 9 I
00000010  41 00 43 00 35 00 54 00 32 00 35 00 48 00 2d 00   A C 5 T 2 5 H -
00000020  36 00 45 00 33 00 32 00 38 00 33 00 42 00 45 00   6 E 3 2 8 3 B E
00000030  34 00 35 00 35 00 35 00 31 00 44 00 44 00 45 00   4 5 5 5 1 D D E
00000040  00 00 6d 00 61 00 69 00 6e 00 2d 00 32 00 00 00     m a i n - 2
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000080  00 00 00 00 00 00 00 00 00
```

Obrázek 5.6: Zachycená komunikace malware

5.4 Využití jiného C&C serveru

Nepřeberné množství různých mutací malwaru CBT-Locker putuje internetem. Při získání funkčních vzorků bylo nutné analyzovat komunikaci. Z analýzy vyplynulo, že jeden ze vzorků využívá vcelku podobný C&C server. Proto se zde naskytla možnost využít tento C&C server. Přesměrování původního malwaru bylo opět provedeno pomocí upravení DNS. Nyní nebyl použit lokální testovací server, ale IP adresa, na kterou směřovala komunikace od nově získaného vzorku. Bohužel, tato metoda opět neuspěla. Komunikace byla jako u vlastního C&C úspěšně navázána, ale ke spuštění šifrovací činnosti malwaru nedošlo.

Při získání funkčního vzorku byl provoz opět analyzován. Z analýzy bylo jasné, že C&C server této mutace je funkční. Při použití HTTPS SSL proxy bylo z výsledků jasné, že se jedná o zcela jinou verzi C&C serveru, přesto byla doména **walkingdead32.ru** přesměrována na IP adresu **193.235.147.102**. Výsledek byl zcela očekávaný, a 1. vzorek přesto nefungoval. Zde jsem další průzkum vzorku zamítl jako směr, který by nevedl k dostatečné detailní analýze malwaru.

5.5 Dynamická analýza

1. vzorek se při testování prokázal jako nefunkční, což by v mnoha ohledech velice komplikovalo detailní analýzu. Především by tato nefunkčnost přinesla komplikace při dynamické analýze, kde by nebylo zcela zřejmé, zda se jedná o nějaký druh obfuskace případně pouze o samotnou nefunkčnost malwaru. Pro úspěšnou analýzu se předpokládá, že zkoumaný vzorek bude na 100 % funkční. Z tohoto důvodu tejdejší vedoucí práce rozhodl, že pro analýzu bude vhodný i jiný vzorek malwaru z rodiny CBT-Locker a analýza na dodaném vzorku se nebude realizovat. U příbuzného malwaru z rodiny CBT-Locker se předpokládalo, že bude funkčně téměř shodný s 1. vzorkem. Může být pouze od jiného autora malwaru. Náročnost obfuskace malwaru se předpokládá stejná nebo vyšší, a to z důvodu, že vývoj malwaru je velice rychlý, a u nově funkčního vzorku se předpokládá, že bude v mnoha ohledech vylepšen a pozměněn.

5.6 Platba Bitcoiny

Od oběti malwaru byla získána adresa virtuální peněženky 1D4S4SNhQ1c c7sApr6eybU7StrqhCiai5b. Tato peněženka byla reálně použita pro převod Bitcoinů, které sloužily jako výkupné za zašifrované soubory, tedy peněženka oběti. Platba byla provedena úspěšně a oběť získala své soubory zpět. Proto je velice vhodné pokusit o detailní analýzu platby. Vzhledem k principům, na kterých je virtuální měna Bitcoin postavena, nelze v detailním rozboru platby předpokládat výraznější odhalení pravé identity autora malwaru.

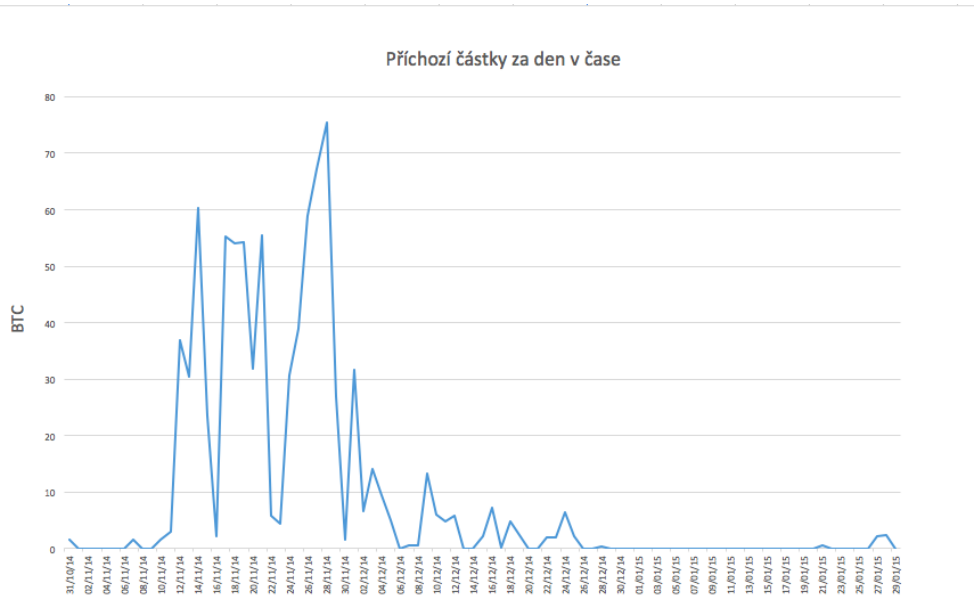
5.6.1 Analýza platby

Pro analýzu platby byla využita webová aplikace na stránce **https://blockchain.info**. Tato webová aplikace umožňuje detailně a přehledně zobrazovat informace o provedené platbě. Z výpisu o platbě je jasně zřetelné, že platba proběhla mezi peněženkami oběti, tedy: **1D4S4SNhQ1cc7sApr6eybU7StrqhCiai5b**, a peněženky zobrazené v informativním okně malwaru: **1FUnE8izy7naYhRtDLEfadobiRxCRZPUS1**. Pokud se nyní zaměříme na peněženku oběti, je z grafu plateb zřetelné, že oběť nakoupila 1.09 BTC, v době uskutečnění tedy asi za 254.51 USD, konkrétně prostřednictvím polské bitcoinové burzy `http://www.simplecoin.pl`. Byla provedena dne 13. 11. 2014 v 15:23:25 hodin. Tato transakce je v Block chainu evidována jako 3a6 1ede8d0cc28c0f55cb5c32e00a248fb34b7a91f6c770f31fd22c9fae6f1c5. Dále můžeme vyčíst z informací zobrazovaných u peněženky oběti odchozí platbu 1c 9254788e5755d19596b82df73e2563742d4d7ff55248a2521ac7a8cde220e v čase 15:23:34 dne 13. 11. 2014. Tato transakce obsahuje dvě položky, první v hodnotě 0.06321064 BTC je poplatek za zpracování transakce v síti Bitcoinu. Druhá obsahuje částku 1.09 BTC. Tato částka byla uvedena na informační stránce malwaru a jednalo se o platbu za odšifrování souborů, proto je vhodné ji dále analyzovat, kam tato částka dále putuje v síti Bitcoin.

5.6.2 Analýza cílové peněženky

Částka 1.09 BTC byla převedena na uvedenou Bitcoin peněženku. Díky webovému nástroji na prohlížení Block chainu můžeme analyzovat všechny pohyby na zmíněné peněžence. Celkově proběhlo přes peněženku 1276 transakcí v přijaté hodnotě 854.48186681 BTC∼217 000 USD. Konkrétně 750 transakcí ve prospěch peněženky a 526 na její úkor. Průběh množství pricházejících Bitcoinů je názorně zobrazen na obr. 5.7. Aktuální stav účtu k datu 7. 10. 2015 je 0 BTC. Pokud projdeme část příchozích plateb na zmíněnou peněženku, často zde najdeme částky v rozmezí okolo 1-3 BTC. Několik desítek těchto připsaných částek na účet by bylo vhodné dále prozkoumat, zda na stejnou adresu nezasílaly výkupné také jiné oběti malwaru. Byla vybrána konkrétně transakce 24e660104772a36ebf3087d3035ecd117f7353a23f84d21bdcab6703798 c11f realizována dne 27. 1. 2015 13:36:18, na peněženku byla připsána částka 2.17074 BTC. Pokud budeme analyzovat původ této částky, zda nepochází z některé z internetových směnáren, setkáme se neúspěchem. Původ z Bitcoin směnárny nebylo možno prokázat. Bohužel i u ostatních analyzovaných příchozích plateb mezi 1-3 BTC nebylo možné identifikovat původ platby, a tedy ani to, zda pochází od obětí malwaru. Dále je nestandardní, že po připsání částky na peněženku o velikosti většinou v zmíněné rozmezí 1-3 BTC následují odchozí transakce. Tato transakce vždy obsahovala 10 odchozích cílů. Příchozí částka se tak vždy odeslala a rozprostřela do 10 menších. Přesné rozvětvení

je názorně ilustrováno na obr. 5.8. Peněženky, na které míří tyto platby, se nikdy neopakovaly. Tento postup můžeme klasifikovat jako postup na zvýšení možnosti skrytí reálné indentity majitele přijímaných Bitcoinů.

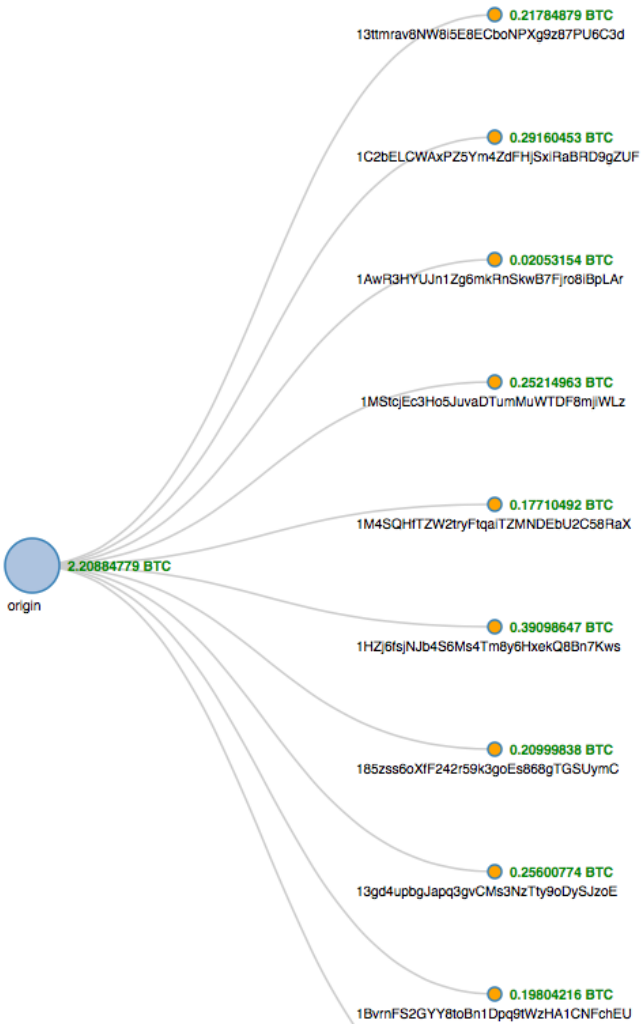


Obrázek 5.7: Příchozí částky Bitcoinu v čase

Nejmladší transakce evidovaná k tomuto datu byla provedena 28. 1. 2014 19:10:56. Jednalo se o odeslání zbývajících BTC na několik různých adres. Naopak první transakce byla realizovaná 31. 10. 2015 12:46:27. Tyto datumy nám pouze potvrzují, že se jedná o peněženku pro příjem Bitcoinů od obětí malwaru. Datumy se shodují s časem kampaně distribující malware.

5.6.3 Zhodnocení analýzy platby

Bohužel, předpoklad, že se nepodaří identifikovat reálnou osobu vlastnící obdržené Bitcoiny, se vyplnil. Tento výsledek se vzhledem k vysoké technické úrovni malwaru dal předpokládat. Autor malwaru nakládá s problematikou Bitcoinu zkušeně. Vidíme to především na tom, že obdržené BTC obratem převádí na jedinečné peneženky.



Obrázek 5.8: Vizualizace rozdělení platby[15]

Kapitola 6

Analýza 2. vzorku

Vzhledem k závěrům v kapitole 5 bylo nutné pro detailnější a objektivnější analýzu získat funkční vzorek malwaru CBT-Locker. Analýza prováděná na nefukčním vzorku může analyzování ztížit či znepřesnit, proto bylo získání funkčního vzorku pro další analýzu velice podstatné.

6.1 Získání vzorku

Obecně je získání funkčního vzorku konkrétního malwaru velice obtížné. O poskytnutí vzorku jsem požádal většinu antivirových firem s pobočkou v České republice. Žádná z 5 oslovených antivirových firem nereagovala. Díky vstřícnému IT oddělení firmy EGP Energoprojekt jsem získal jejich vzorek CBT-Lockeru. Tento vzorek byl ekvivaletní s 1. vzorkem, a tedy nefunkční. Bohužel, osobně jsem tento druh malwaru neobdržel do e-mailové schránky. V prosinci roku 2014 opět probíhal phishingový útok v České republice na internetový obchod jménem **obchod-24.cz**. Tento vzorek malwaru se podařilo získat. Při bližší analýze se ukázalo, že získaný vzorek nebyl žádnou mutací požadovaného malwaru CBT-Locker, ale pouze škodlivým kódem typu trojský kůň. Z tohoto důvodu postrádalo smysl dále získaný vzorek analyzovat. Od příbuzného žijícího v zahraničí se mi podařilo získat funkční vzorek malwaru z rodiny CBT-Locker. V době získání vzorku se jednalo o aktuální škodlivý kód, který byl součástí phishingového útoku v Itálii a Turecku. Díky takto čerstvě získanému vzorku bylo možné pokračovat v analýze malwaru rodiny CBT-Locker.

6.2 Ověření funkčnosti

U nového vzorku bylo nutné nejprve ověřit jeho dostatečnou a plnohodnotnou funkčnost. Toto bylo ověřeno zcela v čistě nainstalovaném operačním systému Microsoft Windows 7 32bit, kde byl vzorek spuštěn. Po chvilce se zobrazilo

okno informující o zašifrování souborů. Pokud jsme prozkoumali souborový systém, nalezli jsme zde zašifrované např. ukázkové obrázky systému Windows. Z tohoto chování bylo zcela zřejmé, že malware je plně funkční a je ho možné plnohodnotně analyzovat.

6.3 Statická analýza

6.3.1 Strings

Na obrázku 6.1 vidíme, že zde nejsou uvedeny žádné smysluplné textové řetězce. Výpis byl získán v programu IDA. Ekvivalentní výpis lze získat i v programu Strings od Sysinternals. Chybějící smysluplné textové řetězce vypovídají o tom, že škodlivý kód s největší pravděpodobností obsahuje nějaký druh obfuskace pomocí šifrování nebo packeru. V analyzovaném kódu se tedy nevyskytuje „jádro“ malwaru – kód v čisté formě, který stojí za opravdovou funkcí malwaru. Analyzovaný vzorek se tedy nejspíše skládá z obálky, která zajišťuje rozbalení, dešifrování a spuštění jádra, které vykonává samotnou činnost malwaru. Tento předpoklad se potvrdí, či vyvrátí při dalším analyzování malwaru.

Address	Length	Type	String
.rdata:00416DA3	00000006	C	:&\bJ>
.rdata:00416DB7	00000005	C	?5L$.
.rdata:00416EAC	00000005	C	+oF>
.rdata:00416F39	00000005	C	\aj\\&\"
.rdata:00416F9B	00000006	C	F\\lE>
.rdata:00417117	00000005	C	?A%My
.rdata:0041717C	00000005	C	f7O>
.rdata:004171C3	00000006	C	B'=>>
.rdata:0041721F	00000005	C	?QNT\t
.rdata:00417284	00000005	C	;,*>
.rdata:004172CB	00000006	C	in]D>
.rdata:004172FB	00000006	C	F\"VM>
.rdata:00417313	00000006	C	30}->
.rdata:0041735C	00000005	C	7`\b>
.rdata:0041744B	00000006	C	0)LK>
.rdata:00417524	00000005	C	DcI>
.rdata:00417734	00000005	C	#tF>
.rdata:00417764	00000005	C	O\n>>
.rdata:004177A7	00000005	C	?43-s
.rdata:004177BF	00000005	C	?P`E5
.rdata:004177F8	00000007	C	1#QNAN
.rdata:00417800	00000006	C	1#INF
.rdata:00417808	00000006	C	1#IND
.rdata:00417810	00000007	C	1#SNAN
.rdata:00417818	0000000B	C	_nextafter

Obrázek 6.1: Textové řetězce obsažené ve funkčním vzorku malwaru

6.3.2 Programové sekce

Z tabulky 6.1 je zřetelně viditelné, že poslední sekce **.rsrc** se vymyká standardu jak velikostí, tak především entropií. Pokud je porovnána zjištěná entropie s průměrnými naměřenými, je zcela jasné, že tato sekce obsahuje jistý typ obfuskace – transformace kódu. Při dynamické analýze bude vhodné se na tento typ obfuskace zaměřit a již na 100 % potvrdit předpoklad z kapitoly 6.3.1.

Tabulka 6.1: Sekce a jejich entropie funkčního vzorku

Název sekce	Entropie[b/B]	Velikost(RAW)[kB]
.text	5.61	23,040
.rdata	5.92	3,072
.rsrc	7.97	691,712

6.3.3 Importy knihoven

Spustitelný soubor malwaru importuje funkce z níže uvedených knihoven:

- KERNEL32.dll – základní operace s WIN32 API
- SHLWAPI.dll – podpora jednoduchých shell nástrojů
- CERTCLI.dll – komunikace a činnost týkající se certifikátů
- MSIMG32.dll – GDIEXT klient
- NDDEAPI.dll – síťová dynamická výměna dat
- USER32.dll – správa oken (vytváření, manipulace), zobrazování oken

Z výše uvedeného výpisu si můžeme povšimnout, že se zde nacházejí obvyklé knihovny (KERNEL32.dll,USER32.dll), které se dají očekávat, ale také méně časté, jako je např. SHLWAPI.dll, CERTCLI.dll. O méně častých knihovnách nelze zcela určitě říci, že se používají k nekalé činnosti, a tím vykazují jistou míru nebezpečnosti.

6.3.4 VirusTotal

Datum první analýzy na VirusTotal nám dává přibližnou informaci o začátku kampaně malwaru. První nahrání na Virus Total proběhlo **28. 1. 2015** v **19:50:36**.

6.3.4.1 Jména souborů

Zde byly díky webové aplikaci VirusTotal získány nejčastější názvy souborů analyzovaného malwaru, které byly nahrány na VirusTotal.

- lwewple.exe
- gxtjkab.exe
- 27055440.exe
- 507346.exe
- nmujsil.exe
- 1149867.exe
- bkzxggm.exe

Z názvů je zřejmé, že jsou náhodné. Ze jmen souborů nelze tudíž odvodit možné programy, za které se malware pokoušel vydávat.

6.4 Dynamická analýza

Pro důkladnou analýzu vlastností a chování funkčního vzorku bylo nutné provést analýzu běžícího malwaru. Běh malwaru byl monitorován několika nástroji a jejich výstup byl následně vyhodnocován. Spuštění funkčního vzorku malwaru vedlo k nevratným změnám v systému, a proto je vhodné před spuštěním malwaru provést pomocí virtualizačního nástroje snapshot. Spuštění malwaru způsobilo vytvoření dvou procesů. 1. proces byl pod jménem spuštěného vzorku malwaru, 2. pod názvem **cvcjxbg.exe**následně byl nalezen injektovaný proces Explorer.exe. Injektovaný proces byl zodpovědný za šifrování souborů.

6.4.1 Změny v souborovém systému

Kromě velkého množství čtení a zápisů na disk šifrovaných souborů zde malware provedl svoje zkopírování do složky
`C:\Users\Honza\AppData\Local\Temp`, soubor pojmenoval **cvcjxbg.exe**.

Kromě tohoto zkopírování a zašifrování vybraných souborů vytvoří malware textový a obrazový soubor ve složce `C:\Users\User\Documents` obsahující veřejný klíč a postup pro zaplacení výkupného. Dále byl ve složce `C:\ProgramData` vytvořen soubor jménem **otzcwtc.html**, ve kterém se nachází HTML tabulka obsahující jméno složky a název zašifrovaného souboru. Soubor je přiložen na CD.

6.4.2 Registry

Malware modifikoval pouze jeden registrový klíč, a to **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**. Tento registr slouží k spuštění příslušného programu, který je uveden v hodnotě registru. Hodnota byla totožná s cestou ke zkopírovanému malwaru.

6.4.3 Komunikace po síti

Komunikace po síti byla monitorována pomocí programu Wireshark. Monitorování sítě bylo aktivováno před samotným spuštěním vzorku malwaru. Při samotném běhu malwaru nebyl detekován žádný provoz. Provoz byl zaznamenán až cca 5 min po zašifrování a zobrazení informačního okna o napadení systému. Z odchyceného provozu (viz soubor tor.pcapng) je zřejmé, že se jedná o komunikace přes síť Tor. Komunikace probíhala na IP adresy: **82.94.251.220**, **104.16.28.16**, **82.94.251.220**, **108.162.232.203**, **37.221.162.226** a **5.135.228.113**. Je zřejmé, že malware pro svůj úspěšný běh nepotřebuje připojení k internetu. Toto jsme si ověřili také pomocí odebrání síťové karty z virtuálního stroje. Běh malwaru byl i přesto úspěšný. Pokud jsme porovnali veřejné klíče, které jsou zobrazovány v informativním okně o zašifrování, byly rozdílné. Z toho můžeme usuzovat, že alespoň některý privátní klíč, který se podílí na šifrování souborů, je generován na napadeném stroji. Dále byla zachycena komunikace na adresu **ip.telize.com**, která slouží k získání veřejné adresy napadeného počítače.

6.4.4 Zaručení perzistence

Malware, aby zaručil svoji perzistenci v systému, zkopíruje sám sebe do složky uvedené v 6.4.1 a následně nastaví do registru, aby byl tento program spouštěn po přihlášení uživatele.

6.5 Získání „jádra“ malwaru pro studium

Při dynamické analýze v OllyDbg prováděl malware několik velice zvláštních operací. V cyklech se pokoušel číst z mnoha souborů, které neexistovaly. Zároveň jména těchto souborů byla zvláštní, které jsou viditelné na obrázku 6.2. Následně se také malware pokoušel číst z neexistujících registrů. Toto chování můžeme připsat snaze autorů malwaru, aby se jejich škodlivý kód na první pohled choval jako standardní neškodný program a také ztížil monitorování pomocí Process Monitorem.

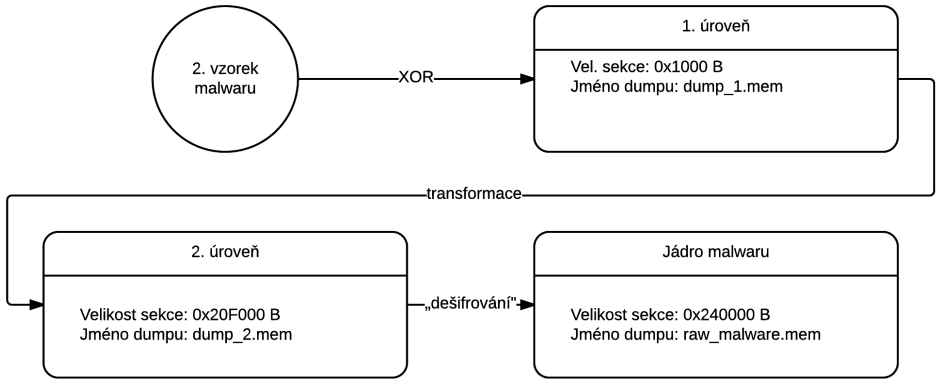
```
.text:00402895                push    10h
.text:00402897                push    offset aBkzsxwnc ; "bKzSXwNC"
.text:0040289C                call    ds:GetFullPathNameW
.text:004028A2                mov     bh, byte_407184
.text:004028A8                mov     byte ptr unk_4070EA, 0F7h
.text:004028AF                and     eax, 38h
.text:004028B2                push    11h
.text:004028B4                push    offset unk_407235
.text:004028B9                push    offset aVnjzwyfeabw ; "VnjZwYfeAbw"
.text:004028BE                call    ds:GetLongPathNameA
.text:004028C4                mov     byte ptr dword_407199, 0D4h
.text:004028CB                mov     byte_40714E, 79h
.text:004028D2                cmp     byte ptr dword_407215, 1
.text:004028D9                jb      loc_40557C
```

Obrázek 6.2: Instrukce pokoušející se získat informace o souboru + špatně analyzovaný kód v programu IDA

6.5.1 Obfuskace

Při analýze kódu nebyla u této verze nalezena žádná funkční obfuskace jako např. isDebuggerPresent a podobné. Bohužel zde byla použita obfuskace pomocí několikanásobného zabalení. Tatovýto kód byl obtížně analyzovatelný v programu IDA. To je zcela zřetelné na obrázku 6.2. Adresy jsou zde obarveny červeně. Červená barva značí v programu IDA neúspěšně analyzovatelný kód. To mělo za následek, že v programu IDA nebylo možno využít všechny jeho schopnosti. S programem OllyDbg se analýza prováděla snadněji.

„Jádro“ malwaru celkově využívá dvě úrovně zabalení. Samotné jádro je zabaleno rutinou, která je sama maskována pomocí binární operace XOR. Každý popis úrovně analyzuje, jak byla úroveň vytvořena a jakým způsobem se přešlo do další úrovně. Poslední stádium je již kód malwaru. Přehled „zabalení“ viz obrázek 6.3.



Obrázek 6.3: Grafický přehled jednotlivých vrstev „zabalení“

Při monitorování procesu před samotným procesem přechodu na první úroveň byly zaznamenány pouze operace **CreateFile** na cestu `C:\Users\honza\Desktop\viry\VnjZwYfeAbw` s výsledkem **NAME NOT FOUND**.

6.5.2 1. úroveň

Pro přechod na první úroveň byly využity tyto instrukce: VirtualAlloc, while cyklus, MemoryProtect, call. Tyto instrukce můžeme nalézt při otevření vzorku malwaru v debuggeru na rozmezí adres 0x004048B9-0x00404A23. Tyto adresy jsou viditelné dále na detailních obrázcích v textu. Při bližší analýze kódu bylo velice jasné, že se jedná o obfuskaci pomocí binární operace XOR. Tato operace je detailněji popsána pomocí následovné funkce $f(x_n)$. Tato obfuskace přímo nerozbalovala kód malwaru, ale pouze další část kódu, která přecházela na další úroveň. Úsek paměti, která byla naalokována, měl velikost 0x00001000 B. Pro první takto transformovaný blok byla použita inicializační hodnota $y_0 = 0x265B24F9$. Tuto hodnotu je možné pozorovat v zeleně ohraničené oblasti na obrázku 6.4, tedy na adrese 0x004048B9.

```
004048B1  . 90              NOP
004048B2  . 90              NOP
004048B3  . 90              NOP
004048B4  . 90              NOP
004048B5  > 31FF            XOR EDI,EDI
004048B7  . 31F6            XOR ESI,ESI
004048B9  . 81EE F9245B26   SUB ESI,265B24F9
004048BF  . F7DE            NEG ESI
004048C1  . BB 16734000     MOV EBX,zpcpreb.00407316
004048C6  . B8 53327260     MOV EAX,60723253
004048CB  . 2D 13327260     SUB EAX,60723213
004048D0  . 50              PUSH EAX
004048D1  . B8 13427260     MOV EAX,60724213
004048D6  . 2D 13327260     SUB EAX,60723213
004048DB  . 50              PUSH EAX
004048DC  . B8 87387260     MOV EAX,60723887
004048E1  . 2D 13327260     SUB EAX,60723213
004048E6  . 50              PUSH EAX
004048E7  . B9 00000000     MOV ECX,0
```

Obrázek 6.4: Vyznačený inicializační hodnota s adresou

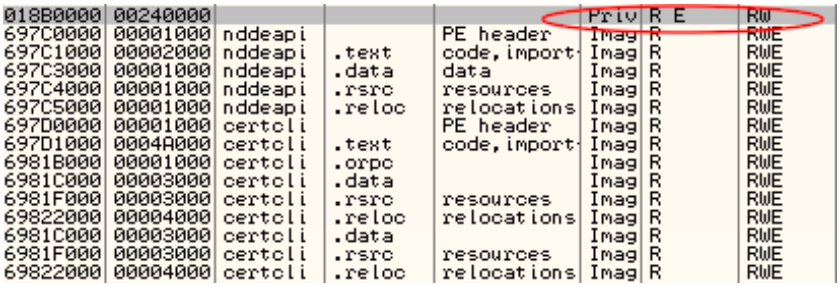
$$y_n = \left[(\overline{x}_n - 0\text{xC}) \oplus ROL(y_{n-1}, 4)\right] + 0\text{x1}$$
$$f(x_n) = ByteSwap(y_n)$$

Následně je zde uveden praktický příklad transformace. Tento příklad obsahuje reálné hodnoty, které byly načteny z paměti.

$$f(0\text{x21FD6D69}) = ByteSwap(y_n)$$
$$y_n = \left[(\overline{0\text{x21FD6D69}} - 0\text{xC}) \oplus ROL(0\text{x89085506}, 4)\right] + 0\text{x1}$$
$$y_n = \left[(0\text{xDE029296} - 0\text{xC}) \oplus 0\text{x90855068}\right] + 0\text{x1}$$
$$y_n = \left[0\text{xDE02928A} \oplus 0\text{x90855068}\right] + 0\text{x1}$$
$$y_n = 0\text{x4E87C2E2} + 0\text{x1}$$
$$y_n = 0\text{x4E87C2E3}$$
$$f(0\text{x21FD6D69}) = ByteSwap(0\text{x4E87C2E3})$$
$$f(0\text{x21FD6D69}) = 0\text{xE3C2874E}$$

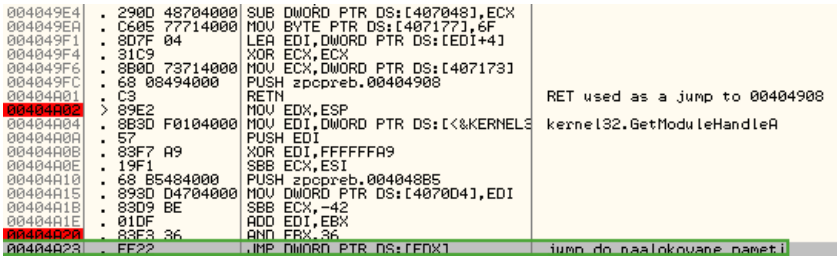
Tedy výsledek funkce $f(0\text{x21FD6D69})$ je 0xE3C2874E. Tato hodnota byla zapsána do předem alokované paměti.

Na obrázku 6.5 vidíme v červeně ohraničené oblasti alokovanou paměť společně s příznaky. Příznak **E** dává možnost spuštění kódu z této sekce. Následně



Obrázek 6.5: Výpis sekcí malwaru s označenou sekcí, kde se nachází rozbalený kód

na obrázku 6.6 je opět v zelené oblasti zřejmé, že instrukce **JMP** skáče do takto alokované paměti, tedy malware přechází na další úroveň. Pro překonání cyklu, který je použit při transformaci a kopírování, je vhodné např. umístit breakpoint na adresu 0x00404A20, tedy jednu instrukci před skokem (viz obr. 6.6). Obsah paměti, do které směřovala instrukce **JMP**, je uložen v souboru **dump_1.mem**.

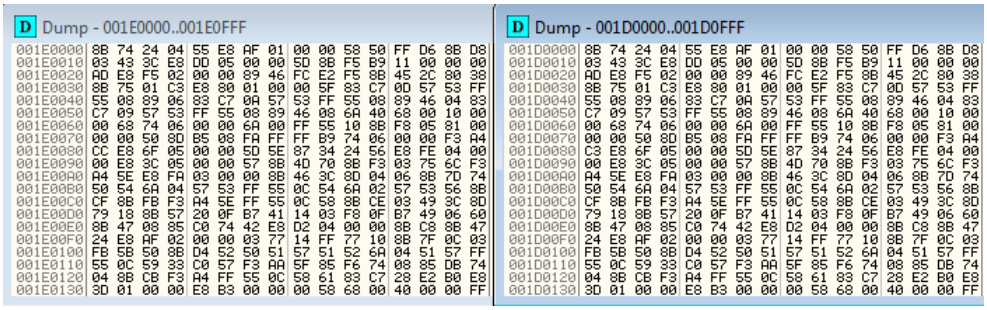


Obrázek 6.6: Zobrazení instrukcí programu před skokem do rozbaleného a naalokovaného programu

Při analýze vykonávaného kódu rozbalené sekce bylo zřetelné, že program zkopíruje rozbalenou sekci identicky (viz obrázek 6.7) do sekce stejné velikosti a začne vykonávat opět kód v nově zkopírované sekci.

6.5.3 2. úroveň

Při přechodu na 2. úroveň byla naalokována paměť velikosti 0x20F000 B. Zde byl obsah nově naalokované paměti vytvářen pomocí instrukcí **LODS** a **STOS**. Přesně byl do sekce zapsán blok o velikosti 0xA22D0 B. Obsah této paměti je uložen v souboru **dump_2.mem**. Pokud se nastaví breakpoint na přístup do paměti na konec zapisovaného bloku, který lze vypočítat ze zmíněné velikosti a počátku naalokované paměti, lze jednoduše obejít cyklus zápisu a pokračovat v analýze prováděného kódu.



Obrázek 6.7: Obsah paměti zobrazující zkopírované sekce

6.5.3.1 Rozbalení kódového „jádra“ malwaru

Při vykonávání kódu 2. úrovně zabalení docházelo k velmi komplexní transformaci načtené hodnoty z programové sekce **.rsrc**, která byla následně zapsána do nově naalokované paměti o velikosti 0x240000 B. Vzhledem k entropii, která byla vypočítána pro tuto programovou sekci **.rsrc**, zde byl vysloven předpoklad pro výskyt jisté obdoby šifrování. Tento předpoklad se vyplnil a načtená hodnota procházela transformací. Výsledek této transformace je následně zapisován do naalokované paměti. Detailní popis transformace pro další analýzu malwaru nebyl nutný.

Na obrázku 6.8 je zcela z obsahu paměti zřetelné, že malware dokončil své rozbalení. Tomu nasvědčovalo několik faktorů. Především bylo při zobrazení této sekce zřetelné, že paměť sekce obsahuje jednotlivé textové řetězce, které jsou zobrazovány v okně, jež je zobrazeno při úspěšném běhu malwaru. Z této sekce paměti bylo vhodné vytvořit dump. Tento dump je uložen v souboru **raw_malware.mem**. Dump obsahuje kompletní kód malwaru, tedy kódové jádro, které je zodpovědné za veškerou škodlivou činnost malwaru. Následně program přešel do rozbalené sekce pomocí instrukce **call**. Tato instrukce se nachází na adrese 0x15F11CB, pokud naalokovaná sekce začíná na adrese 0x1550000.

Takto získaný kód není možné ihned analyzovat, a to z důvodu, že např. adresy uvedené v IAT neodpovídají adresám, které by měla tabulka obsahovat.

6.5.3.2 Problematika rozbalení malwaru

Při překonávání cyklů, které jsou použity v jednotlivých úrovních, byl využit postup, kdy bylo při prvním běhu sledováno, kde konkrétně rozbalování v segmentu paměti skončí. Tuto adresu bylo vhodné si poznamenat. V tento moment byl vrácen snapshot do stavu před rozbalováním a nastaven hardwarový breakpoint na přístup na adresu, která byla získána v předešlém běhu. Běh programu se zde zastavil. V tomto místě se již ocitl za cyklem, který byl zodpovědný za celkové rozbalení. Dále bylo velmi nutné věnovat zvýše-

nou pozornost jednotlivým instrukcím **call**. Pokud se instrukce přehlédla a zvolil se v debuggeru krok „Step over“ a nebyl vhodně nastavený breakpoint v programu, což znamená, že utekl, nebylo možné již prováděné úkony řádně analyzovat. V těchto problémových místech bylo vhodné použít v debuggeru „Step into“ debugger zde skočil do požadované sekce prováděného kódu.

```
01A1DF90 2C 00 00 00 2A 3A 38 30 00 00 00 00 2A 3A 34 34 ,...*:80....*:44
01A1DFA0 33 00 00 00 72 65 6A 65 63 74 20 2A 3A 2A 00 00 3...reject *:*..
01A1DFB0 72 65 6C 61 79 00 00 00 55 6E 72 65 63 6F 67 6E relay...Unrecogn
01A1DFC0 69 7A 65 64 20 76 61 6C 75 65 20 27 25 73 27 20 ized value '%s'
01A1DFD0 69 6E 20 53 61 66 65 4C 6F 67 67 69 6E 67 00 00 in SafeLogging..
01A1DFE0 55 6E 72 65 63 6F 67 6E 69 7A 65 64 20 76 61 6C Unrecognized val
01A1DFF0 75 65 20 69 6E 20 50 75 62 6C 69 73 68 53 65 72 ue in PublishSer
01A1E000 76 65 72 44 65 73 63 72 69 70 74 6F 72 00 00 00 verDescriptor...
01A1E010 50 61 74 68 42 69 61 73 4E 6F 74 69 63 65 52 61 PathBiasNoticeRa
01A1E020 74 65 20 69 73 20 74 6F 6F 20 68 69 67 68 2E 20 te is too high.
01A1E030 49 74 20 6D 75 73 74 20 62 65 20 62 65 74 77 65 It must be betwe
01A1E040 65 6E 20 30 20 61 6E 64 20 31 2E 30 00 00 00 00 en 0 and 1.0....
01A1E050 50 61 74 68 42 69 61 73 57 61 72 6E 52 61 74 65 PathBiasWarnRate
01A1E060 20 69 73 20 74 6F 6F 20 68 69 67 68 2E 20 49 74  is too high. It
01A1E070 20 6D 75 73 74 20 62 65 20 62 65 74 77 65 65 6E  must be betwen
```

Obrázek 6.8: Obsah paměti po rozbalení

Výše zmíněný postup se z popisu zdá velice snadný. Bohužel, překonání a získání dumpu nebylo takto snadné a před touto úspěšnou metodou bylo nutné projít několik metod, které ve výsledku nebyly úspěšné, jako je např. získání jádra malwaru z dumpu RAM paměti uspaného virtualizovaného operačního systému.

6.6 Rekonstrukce malwaru pro studium

V předchozí kapitole bylo vysvětleno, jak překonat veškeré obfuskace zakomponované do analyzovaného malwaru, neboť získaný kód není možné spustit a provádět na tomto vzorku analýzu. Bylo nutné takto získaný dump paměti zrekonstruovat. Tato rekonstrukce byla velice náročná, protože získaný dump neobsahoval PE hlavičku. Neobsažení PE hlavičky podle poznatků o ostatních malwarech není zcela častá vlastnost a vyskytuje se pouze sporadicky. Její nepřítomnost velice ztížila rekonstrukci celého spustitelného programu. Absenci hlavičky nelze nazvat zcela jistě obfuskační metodou. Její nepřítomnost je možné připsat změně packeru při vývoji této konkrétní verze. Velice důležité bylo, že při bližším zkoumání dumpu v hex editoru nebyla nalezena relokační sekce. Její nepřítomnost byla pozitivní, a to z důvodu, že ulehčí následnou rekonstrukci.

Zde jsem úspěšně použil nástroj ImpREC. Tento nástroj umožňuje rekonstruovat IAT[14] z dostupného běžícího procesu. Z důvodu, že dump měl adresy v tabulce IAT neplatné, bylo nutné použít tento nástroj pro jejich obnovu. Na obrázku 6.9 je zřetelná sekce **.empy**. Tuto sekci přidal program ImpREC a slouží k opravě IAT tabulky.

[14]import address table-tabulka pou

Property	Value	Value	Value	Value
Name	.empy	.text	.rdata	.mackt
Virtual Size (bytes)	0x002C0000 (2883584)	0x000E4000 (933888)	0x0015F000 (1437696)	0x00002000 (8192)
Virtual Address	0x00001000	0x002C1000	0x003A5000	0x00504000
Raw Size (bytes)	0x002C0000 (2883584)	0x000E4000 (933888)	0x0015F000 (1437696)	0x00002000 (8192)

Obrázek 6.9: Programové sekce opraveného dumpu

Tento způsob rekonstrukce byl realizován ve spolupráci s Katedrou počítačových systému ČVUT FIT, konkrétně s Ing. Tomášem Zahradnickým, EUR ING, Ph.D. a Ing. Martinem Jirkalem. Po úspěšné rekonstrukci bylo opět důležité ověřit funkčnost získaného „surového“ malwaru. Rekonstruovaný malware se ukázal jako funkční a mohl jsem přikročit k detailní analýze.

6.7 Analýza rekonstruovaného jádra malwaru

Díky získání neobfuskovaného malwaru jsem mohl zahájit samotnou analýzu. Předešlé kroky, které vedly k získání malwaru, byly znalostně a časově náročné. Následná analýza pokračovala známými postupy, které byly použity v předešlých kapitolách při analýze. Program IDA již tento škodlivý kód správně analyzoval a v programu byly viditelné jednotlivé basic bloky.

6.7.1 Statická analýza

6.7.1.1 Strings

Při otevření rekonstruovaného jádra malwaru v hex editoru jsou velice dobře zřetelné textové řetězce (viz obr. 6.10) obsažené v malwaru. Tyto textové řetězce si můžeme detailně a přehledně zobrazit v programu IDA v subview Strings. Především si můžeme všimnout, že malware je napsán v několika jazykových mutacích. Dále bylo z textových řetezců zřejmé, že program používá anonymizační program Tor a pro platbu výkupného virtuální měnu Bitcoin.

's'	.rdata:007B40D4	00002DA0	C	netilnedvlse<screentext> %a1%%f3%%c3%I tuoi dati personali sono criptati da CTB-
's'	.rdata:0081925C	00002D94	C	<screentext> %a1%%f3%%c3%I tuoi dati personali sono criptati da CTB-Locker.%f0%
's'	.rdata:007FACF4	00002D94	C	<screentext> %a1%%f3%%c3%I tuoi dati personali sono criptati da CTB-Locker.%f0%
's'	.rdata:0081341C	00002AFB	C	<screentext> %a1%%f3%%c3%Your personal files are encrypted by CTB-Locker.%f0%
's'	.rdata:007F4EBC	00002AFB	C	<screentext> %a1%%f3%%c3%Your personal files are encrypted by CTB-Locker.%f0%
's'	.rdata:007AE2A0	00002AFB	C	<screentext> %a1%%f3%%c3%Your personal files are encrypted by CTB-Locker.%f0%
's'	.rdata:0082031F	00001BC0	C	enam nav iespējas atšifrēt jūsu failus tikmēr, kamēr jūs nesamaksāsiet prasīto summu
's'	.rdata:00800000	000008BD	C	kunt Bitcoins kopen met contant geld, elektronisch geld, directe overschrijving, prep
's'	.rdata:007B3906	000007CD	C	er Nähe verkauft.\r\n\r\nKaufen %btcprice% BTC (etwa von %eurprice% EUR) und st

Obrázek 6.10: Textové řetězce obsažené v malwaru

6.7.1.2 Importy knihoven

Na obrázku 6.11 je viditelné, že rekonstruované jádro malwaru obsahuje zcela identické importy knihoven jako analyzovaný 2. vzorek malwaru (viz kapi-

tola 6.3.3).

Library (6)	Blacklisted (3)	Type	Symbols (65)	Description
certcli.dll	x	Implicit	5	Microsoft® Active Directory Certificate Services Client
msimg32.dll	x	Implicit	3	GDIEXT Client DLL
nddeapi.dll	x	Implicit	3	Network DDE Share Management APIs
user32.dll	-	Implicit	16	Multi-User Windows USER API Client DLL
shlwapi.dll	-	Implicit	12	Shell Light-weight Utility Library
kernel32.dll	-	Implicit	26	Windows NT BASE API Client DLL

Obrázek 6.11: Importy rozbaleného malwaru

6.7.2 Analýza v programu IDA

Díky rekonstrukci již byla IDA schopna analyzovat kód a zobrazila jednotlivé basic blocky programu. Malware v této podobě byl již zbaven veškerých obfuskací a byl v programu dobře čitelný. Nalezené zajímavé funkce jsem označil názvy s prefixem **R**. Jejich seznam lze zobrazit v programu IDA v subview Functions. Soubor, který byl vytvořen analýzou „jádra“ malwaru v programu IDA , je uložen pod názvem **malware.idb**. V textu je zmíněna pouze část rozpoznaných funkcí.

6.7.2.1 Generování klíče

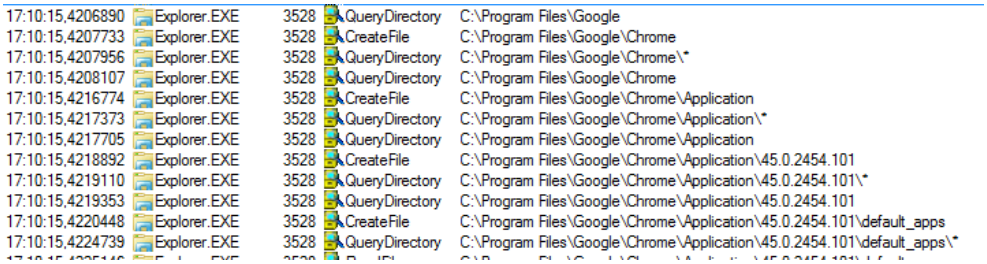
Generování klíče je podstatnou částí každého šifrovacího mechanismu. Pokud by byl klíč generován nevhodným a předvídatelným způsobem, ohrozil by samotnou odolnost šifry před útoky. Při analýze malwaru bylo pozorováno velice zdařilé generování klíče (viz ukázka kódu). Na této ukázce kódu vidíme základ pro generovaný klíč, který sbírá z vygenerování náhodnosti systémového času, počtu ticků od spuštění počítače, ID vláken a ID procesu. Takto získaných 52 bytů je následně vloženo do hashovací funkce SHA256. Tuto funkci lze nalézt v souboru **malware.idb** jako funkci pojmenovanou jako **R__gen__key** na adrese 0x0074E13D.

```
RandBPool = alloc();
RandBPool.random = CryptGenRandom(0, 0x14u, );
RandBPool.systemTime = GetSystemTimeAsFileTime();
RandBPool.tickCount = GetTickCount();
RandBPool.threadId = __ROL4__(GetCurrentThreadId(), 16);
RandBPool.other = RandBPool.tickCount ^ GetCurrentProcessId();
SHA256(&RandomBytePool, sizeof(RandBPool))
```

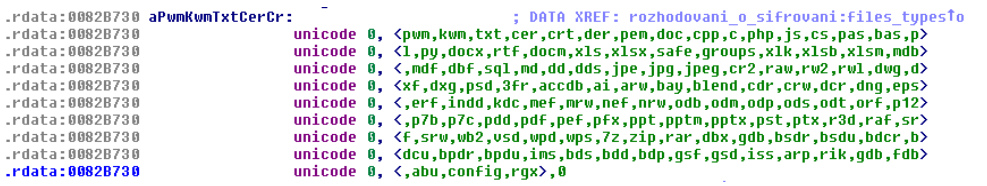
6.7.2.2 Soubory, které budou zašifrovány

Malware nešifruje úplně všechny soubory, které jsou na disku přítomny, ale pouze soubory vybraných typů. Přípony jsou viditelné na obrázku 6.13 a jsou dohledatelné na adrese 0x0082B730. Tyto typy útočníci vybrali záměrně, a to

z důvodu, že soubory bývají nejcennější a obsahují většinou velice hodnotné informace pro uživatele, což se o systémových souborech nedá říci. Pokud by malware šifroval úplně celý disk, ohrozil by samotnou funkčnost operačního systému, a to autoři samozřejmě nechtějí. Jejich cílem je získat od uživatele poškozeného počítače finanční částku. Na obrázku 6.12 je zcela zřetelný sestup v souborovém systému napadeného počítače.



Obrázek 6.12: Výpis aktivity malwaru při procházení souborovým systémem napadeného operačního systému



Obrázek 6.13: Paměť programu obsahující přípony souborů

6.7.2.3 Šifrování souborů

Malware používá pro šifrování souborů algoritmus AES společně s asymetrickým šifrováním algoritmem D-H na eliptických křivkách „(ECDH)“. Každý soubor je zašifrován jedinečně vygenerovaným klíčem. Následně je použit ECDH. Bohužel se nepodařilo rekonstruovat přesný mechanismus šifrování. Byly pouze nalezeny fragmenty. Mezi tyto fragmenty můžeme zařadit pojmenované funkce: **R__crypt__file**, **R__crypto__context**, **R__dh__generator**.

6.7.2.4 Nastavení pozadí

Zde bohužel analýza selhala a nebylo nalezeno volání, které nastaví aktuální pozadí plochy. Soubor, který je umístěn na pozadí plochy, se nachází ve složce `C:\Users\User\Documents`. Nastavení pozadí plochy se předpokládá zavoláním pomocí **SystemParametrsInfo**. Tato funkce je obsažena v analyzovaném malwaru, ale bohužel se nepodařilo přesně lokalizovat použití této funkce.

```
.rdata:007EC0EC aSystemparamete db 'SystemParametersInfoW',0
.rdata:007EC102                 align 4
```

Obrázek 6.14: Funkce pro nastavení pozadí napadeného počítače

6.7.2.5 Informační rozhraní s uživatelem napadeného počítače

Malware informuje uživatele o úspěšném napadení několika způsoby. Za prvé změní pozadí obrazovky a na pozadí zobrazí potřebné informace. Za druhé minimalizuje všechna aktivní okna a zobrazí nové okno. V tomto okně jsou opět uvedeny informace o zašifrování a možném rozšifrování. Poskytuje také informaci o časové lhůtě pro možné zaplacení výkupného. Následně jsou uvedené informace shrnuty v textovém souboru ve složce, ve které je umístěn obrázkový soubor, který je zobrazen na pracovní ploše.

6.7.2.6 Zobrazení seznamu zašifrovaných souborů

Malware umožňuje v informativním okně zobrazit tabulku. Tabulka obsahuje seznam cest a názvů zašifrovaných souborů. Při analýze se podařilo funkci, která je zodpovědná za vytváření tabulky, identifikovat. Na obrázku 6.15 můžeme pozorovat HTML tag, který je při vytváření tabulky použit. Pro samotné vytváření tabulky je využito samostatné vlákno. K vytváření tabulky dochází zároveň se šifrováním souborů.

```
.text:007537F9                  mov     esi, offset aTdTd ; "</td><td>"
```

Obrázek 6.15: HTML tag použitý při generování tabulky zašifrovaných souborů

6.7.2.7 Odšifrování čtyř souborů

Informativní okno nabízí pro zvýšení důvěry oběti zdarma rozšifrování čtyř vybraných souborů. Bohužel se nepodařilo přesně lokalizovat funkci a princip odšifrování. Toto místo lze považovat za vhodné pro budoucí analýzy malwaru CBT-Locker.

6.7.2.8 Obrana proti virtualizaci a antivirům

Na kódu je jasně zřetelné, že malware kontroluje, zda běží procesy avp.exe (Kaspersky Antivirus), ekrn.exe (ESET nod32), vboxtray.exe (Virtual Box), vboxservice.exe (Virtual Box), vmtoolsd.exe (VMWare). První dva jsou antivirové programy a zbylé jsou virtualizační. Z toho vyplývá, že malware zjišťuje, zda neběží ve virtualizovaném prostředí. O účelu kontroly antivirů se může pouze spekulovat, vzhledem k tomu, že firmy ESET a Kaspersky cíleně bojují proti tomuto druhu malwaru. Tvůrci malware se mohou touto obfuskací bránit

brzkému odhalení ze strany antivirů, a tím pádem omezit nutnost zabezpečování a znovuanalyzování nové schopnosti odhalování antiviru. Kód zodpovědný za tuto ochranu můžeme nalézt ve funkcích zvaných **R__check__antivir** a **R__check__vm**.

```
lea     eax, [ebp+pe.szExeFile]
push    offset aAvp_exe ; "avp.exe"
push    eax             ; wchar_t *
....
....
lea     eax, [ebp+pe.szExeFile]
push    offset aEkrn_exe ; "ekrn.exe"
push    eax             ; wchar_t *
....
....
push    eax             ; lppe
push    esi             ; hSnapshot
call    ds:Process32NextW
```

```
push    offset aVboxtray_exe ; "vboxtray.exe"
push    eax             ; wchar_t *
....
....
jz      short loc_74E9F8
lea     eax, [ebp+pe.szExeFile]
push    offset aVboxservice_ex ; "vboxservice.exe"
push    eax             ; wchar_t *
....
....
jz      short loc_74E9F8
lea     eax, [ebp+pe.szExeFile]
push    offset aVmtoolsd_exe ; "vmtoolsd.exe"
push    eax             ; wchar_t *
```

6.7.2.9 Postřehy z kódu

Jak obálka, tak samotný malware nepoužívaly ASLR. Dále bylo zvláštní, že samotná obálka „jádra“ neobsahovala žadné základní obfuskace, jako je např. isDebuggerPresent a podobné. To autoři malwaru vynahradili několikánásobným „zabalením“. Také je možné se domnívat, že autoři vynechali tyto základní obfuskace záměrně, aby dropper pro běžného pozorovatele vypadal jako běžný a standardní program.

6.7.2.10 Získání IP adresy

Pro vybudování Tor proxy v systému potřebuje malware zjistit IP adresu napadeného počítače na internetu (viz obrázek 6.16). Dle analýz zdrojového kódu a komunikace využívá malware službu telize.com. Se službou komuni-

kuje přes protokol HTTP a z tohoto důvodu je zachycená komunikace zcela jednoduše čitelná. Následně je zde při reverzní analýze kódu opět objevena, konkrétně v pojmenované funkci **R_get_IP**. V kódu je zobrazena zcela kompletní HTTP metoda GET s jednotlivými URL parametry. Následně získaná veřejná IP adresa je použita ve vytváření Tor proxy. Tor proxy je následně využívána v malwaru.

```
getaddrinfo("ip.telize.com", "80", &pHints, &ppResult)
```

Obrázek 6.16: Získání IP adresy

6.7.2.11 Jazyková mutace

Z analýzy textových řetězců, které zobrazil program String i IDA, bylo zřetelné, že malware obsahuje několik jazykových mutací, konkrétně angličtinu, němčinu, španělštinu, italštinu, francouzštinu a lotyštinu. Malware jako výchozí jazyk zobrazuje anglický jazyk. Malware neobsahuje žádné funkce na zjištění systémového nastavení.

6.7.3 Porovnání 1. vzorku a 2. vzorku malwaru

Z hlediska sítě lze pozorovat mnoho rozdílů. 1. vzorek komunikoval přes HTTPS protokol. Detailně analyzovaný vzorek pro tuto komunikaci využívá anonymizační síť Tor. Tento posun ve vývoji je možno připsat zhruba pětiměsíčnímu časovému rozdílu mezi vydáním jednotlivých verzí malwaru. Během tohoto času mohl malware prodělat značný vývoj a pozměnit tedy svoji strukturu a vlastnosti. Mohl také reagovat na obranné prostředky ze strany antivirových společností a upravit své chování takovým způsobem, aby byl složitěji odhalitelný.

6.7.3.1 Srovnání z hlediska programového překladače

Z hlediska struktury kódu 1. vzorek obsahoval základní obfuskace a v oblasti sekcí nebyly nijak výrazné entropie jako v případě analyzovaného vzorku. 1. vzorek obsahoval základní ochrany zásobníku Buffer Security Check pomocí tzv. kanárků. Tato vlastnost nebyla u funkčního vzorků detekována.

6.7.3.2 Srovnání z hlediska struktury

1. vzorek neobsahoval ani náznakem podobný dropper, jako byl pozorován u funkčního vzorku.

6.7.3.3 Srovnání z hlediska uživatelského rozhraní

1. vzorek malwaru byl lokalizován pouze pro Českou republiku. Analyzovaný funkční vzorek český jazyk neobsahoval.

6.7.4 Skrývání před antiviry

Ve většině malwaru má obfuskace za úkol ztížení jeho analýzy a následně detekce pomocí jednotlivých heuristik. Zde ve vzorku číslo 2 jsme se o tom mohli detailně přesvědčit.

Vyplývá to jak z monitoringu zapisování na souborový systém, tak ze sledování činnosti nad registry.

Bylo zcela zřetelné, že tyto úkony v dropperu neplní absolutně žádnou smysluplnou funkci a pouze se tímto chováním se pokouší skrýt svůj pravý účel činnosti. Těmito úkony se snaží analyzovaný malware vzbudit pro pozorovatele, jako je např. antivirový program, dojem, že se jedná o standardní program nevykonávající žádnou nebezpečnou činnost. Tato obfuskace poskytuje ještě jednu velkou výhodu, a to, že je velice snadné měnit kód dropperu a samotný škodlivý kód nemusí být nijak více modifikován. To umožňuje tvůrci malwaru strojově generovat další mutace malwaru. Tomuto se snaží antivirové společnosti bránit díky popsání pomocí genetiky. Toto popsání malwaru následně umožňuje odhalovat infikované soubory.

6.8 Ochrana před CBT-Lockerem

Ochrana před jedním druhem CBT-Lockeru by byla velice snadná. Díky postupům, které používá a kam zapisuje, by jeho činnost bylo možné překazit v inicializační části. Bohužel, tvůrci malwaru jsou si této zranitelnosti dobře vědomi a každým dnem vydávají nové a nové verze, které se mnohdy velmi liší. I pro samotné antivirové firmy je obtížné této hrozbě účinně čelit. Nejslabším článkem v této věci je opět sám uživatel. Pokud uživatel bude důvěřovat neznámým e-mailům a přílohám v nich obsaženým, bude vždy obtížné zabránit nákaze. Nejlepší ochranou jsou známá pravidla, která jsou stále zdůrazňována, ale bohužel běžní uživatelé tato pravidla nedodržují. Mezi pravidla patří pravidelné zálohování dat, aktualizovaný operační systém, antivirový program a především zdravý úsudek při otevírání příloh. Tyto zásady bezpečného chování pomohou uživatelům vyhnout se zmíněným nákazám.

6.9 Zhodnocení reverzní analýzy

Autoři malwaru jsou ve vytváření škodlivých kódů nápaditi a vynalézaví. Využívají mnoho způsobů nejrůznějších obfuskací. Velmi proměnlivě mění C&C servery a skoro každý den vydávají nové verze malwaru. Proto zde bylo náročné zorientovat se v použitých technikách obfuskace a tyto jednotlivé

ochrany překonat. Při zpracování této práce se ukázalo, že reversing malwaru je v oboru reverzního inženýrství jedním z nejnáročnějších témat. Už samotné získání škodlivého kódu a jeho zpětnou rekonstrukci vhodnou pro studium bylo velice časově náročné.

6.10 Zhodnocení nebezpečnosti

CBT-Locker můžeme zařadit mezi nejzrádnějším druhy malwaru, a to nejenom jeho zákeřnost, ale také účinnost a psychickým nátlak. Jeho velká účinnost je také daná velikostí působnosti. Dále je jeho nebezpečnost daná velkou úrovní použitých komponent, jako je např. mírou obfuskace atd.

- Silného šifrování – v analyzované mutaci bylo použito šifrování AES a eliptické křivky. Toto šifrování je velice silné a jeho prolomení nemožné v reálném čase, kdy jsou zašifrovaná data ještě relevantní.
- Anonymní platby – důležitost anonymní platby již byla zmíněna v kapitole 2.2. Při analýze malwaru bylo ověřeno, že tyto principy používá.
- Použití Tor – malware používá projekt Tor v mnoha ohledech, především pro kumunikaci mezi obětí a autorem malwaru. Použití této anonymizační sítě zvýšilo anonymitu autora a nebezpečnost malwaru.
- Malá velikost samotného malwaru – soubor stahovaný z phishingových stránek měl velikost cca 770 kB. Tato velikost vzhledem k dnešní velikosti souborů, např. různých faktur od internetových obchodů atd. nepřesahovala velikost, se kterou jsou běžní uživatelé naučeni pracovat. Pokud by velikost škodlivého kódu přesahovala obvyklou velikost souborů, bylo by možná množství napadených počítačů o několik desítek procent menší.
- Technická úroveň – malware vzhledem k použitým technologiím a metodám obfuskace na velmi vysoké technické úrovni.
- Uživatelské přívětivosti – malware poskytuje možnost volby jazyka, který uživateli vyhovuje. Má na výběr celkem z šesti jazyků.

6.11 Placení výkupného ano/ne?

Na internetových fórech zabývajících se problematikou zašifrování malwaru CBT-Locker se často varuje, aby oběti CBT-Lockeru neplatily výkupné. Na tuto otázku je nutné se podívat ze dvou pohledů. Nejdříve z pohledu oběti. Zde je zřejmé, že pokud uživatel nemá zašifrovaná data řádně zazálohovaná a data jsou pro něj životně důležitá, nic jiného než zaplatit bohužel nezbývá.

Odborníci na počítačovou bezpečnost často zastávají názor, že by oběti výkupné neměly platit. Tuto volbu prosazují z zcela jasných důvodů. Pokud

autoři malwaru nebudou mít dostatečný příjem z vytvářeného malwaru, jejich motivace zlepšovat a vyvíjet bude menší a hrozby ze strany malwaru ubude. Bohužel malware může mít několik účelů. Kampaň, která se nám může jevit, že má za cíl obohacení autorů, může být cílená pouze na jednu osobu a jedna konkrétní data. Masivní kampaní získá krytí pro tuto specifickou akci. Dle slov expertů z ESETu je tato praxe vcelku častá. Ano, pokud jsou data alespoň částečně nahraditelná, je vhodné výkupné v žádném případě neplatit.

Závěr

Reverzní inženýrství představuje velmi obsáhlý obor, kvůli kterému jsem byl přinucen nastudovat široké spektrum informací z různých zdrojů. Tato problematika je v dnešní digitální době velice aktuální a hrozba kyberzločinu stále stoupá. Rizika z toho plynoucí se týkají nejenom mého odborného zaměření v informačních technologiích, ale také běžné populace, protože informační technologie vstupují určitou formou do života každého člověka.

Zadaný vzorek malwaru nebyl funčkní. Získaní funkčního vzorku malwaru z rodiny CBT-Locker bylo velmi náročné.

Následně jsem různými metodami analyzoval mutaci malwaru CBT-Locker. Samotná analýza byla velice obtížná, protože malware CBT-Locker můžeme zařadit mezi špičku malwaru. Jeho tvůrci jsou ve svém oboru experti, používají nejnovější technologie a stále svůj kód zdokonalují. Přístup k nejnovějším informacím ohledně technik malwaru je velice obtížný z důvodu, že autoři malwaru tyto informace drží v tajnosti a z důvodu obtížnější obrany vůči nim antivirové firmy informace neposkytují, protože se jedná o část jejich know-how. O jejich kvalitách svědčí také to, že přes hrozbu, jakou CBT-Locker představuje, a přes snahu velkých antivirových firem jeho autoři nebyli doposud odhaleni, v důsledku toho mohou stále páchat velké škody na cenných datech.

Z analýz jsem zjistil chování malwaru CBT-Locker a základní metody, které používá. Práce na analýzách mi přinesla mnoho poznatků, které mě uvedly do této problematiky, a mohl jsem popsat řadu detailů o chování CBT-Lockeru. Doufám, že znalosti získané při vypracování této práce úspěšně přenesu na mobilní platformu, tedy reverzní inženýrství aplikací pro mobilní operační systémy iOS a Android. V této oblasti je bezpečnost často podceňována vzhledem k tomu, jaké informace jsou svěřovány chytrým mobilním zařízením.

Literatura

[1] Viri.cz: Sledování zásilky České pošty aneb nová havěť [online]. Listopad 2014, [Citováno 10. 10. 2015]. Dostupné z: http://www.viry.cz/sledovani-zasilky-ceske-posty-aneb-nova-havet/

[2] Česká pošta: Upozornění pro firmy–další phisingový útok [online]. Listopad 2014, [Citováno 5. 11. 2015]. Dostupné z: https://www.ceskaposta.cz/-/upozorneni-pro-klienty-dalsi-phishingovy-utok

[3] Roman, V.: Útočníci se tváří jako Česká pošta, aby vám napadli PC. Pozor na e-maily [online]. Listopad 2014, [Citováno 5. 11. 2015]. Dostupné z: http://technet.idnes.cz/falesna-zprava-ceska-posta-d0z-/sw_internet.aspx?c=A141113_165038_sw_internet_vse

[4] Česká pošta [online]. [Citováno 1. 1. 2015]. Dostupné z: https://www.postaonline.cz/trackandtrace

[5] TOR (The Onion Router)[online]. Leden 2015, [Citováno 11. 10. 2015]. Dostupné z: http://www.cs.sit.kmutt.ac.th/blog/?p=142

[6] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System, 2008, [Citováno 10. 10. 2015]. Dostupné z: https://bitcoin.org/bitcoin.pdf

[7] Nathan, J.: A brief history of bitcoin – and where it's going next [online]. Březen 2015, [Citováno 10. 10. 2015]. Dostupné z: http://thenextweb.com/insider/2015/03/29/a-brief-history-of-bitcoin-and-where-its-going-next/

[8] Netmarketshare.com Desktop share by version [online]. Listopad 2014, [Citováno 15. 10. 2015]. Dostupné z: http://netmarketshare.com

[9] Eagle, C.: *The IDA pro book*. San Francisco: No Starch Press, druhé vydání, 2011, ISBN 1593272898.

[10] Eilam, E.; Chikofsky, E. J.: *Reversing: Secrets of reverse engineering.* Indianapolis: Wiley, 2005, ISBN 978-076-4574-818.

[11] Lukan, D.: Linear Sweep vs Recursive Disassembling Algorithm [online]. Únor 2013, [Citováno 5. 11. 2015]. Dostupné z: http://resources.infosecinstitute.com/linear-sweep-vs-recursive-disassembling-algorithm/

[12] VirusTotal. Listopad 2014, [Citováno 15. 10. 2015]. Dostupné z: https://www.virustotal.com/en/about/

[13] Collberg, C.; Thomborson, C.; Low, D.: A Taxonomy of Obfuscating Transformations[online]. 1997/2009, [Citováno 15. 10. 2015]. Dostupné z: https://researchspace.auckland.ac.nz/bitstream/handle/2292/3491/TR148.pdf

[14] Schiffman, M.: Cisco malware history [online]. Únor 2010, [Citováno 5. 11. 2015]. Dostupné z: http://blogs.cisco.com/security/a_brief_history_of_malware_obfuscation_part_1_of_2

[15] Blockchain info [online]. [Citováno 5. 11. 2015]. Dostupné z: https://blockchain.info

Příloha **A**

Seznam použitých zkratek

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol Secure

IP Internet Protocol

C&C Command and Control

PHP Hypertext Preprocessor

IAT Import address table

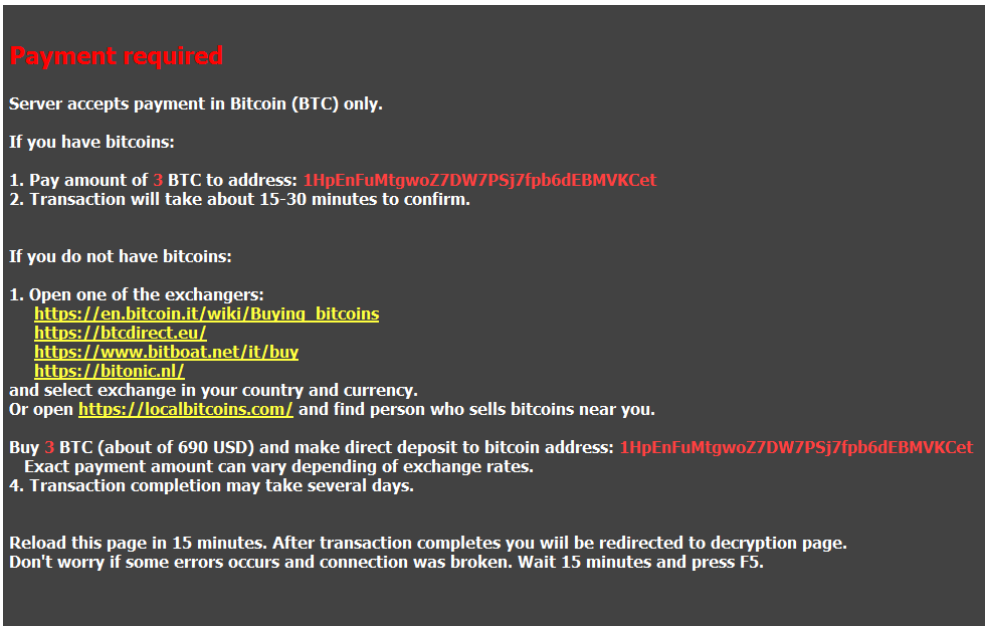
PE Portable executable

DDOS Distributed Denial of Service

Příloha B

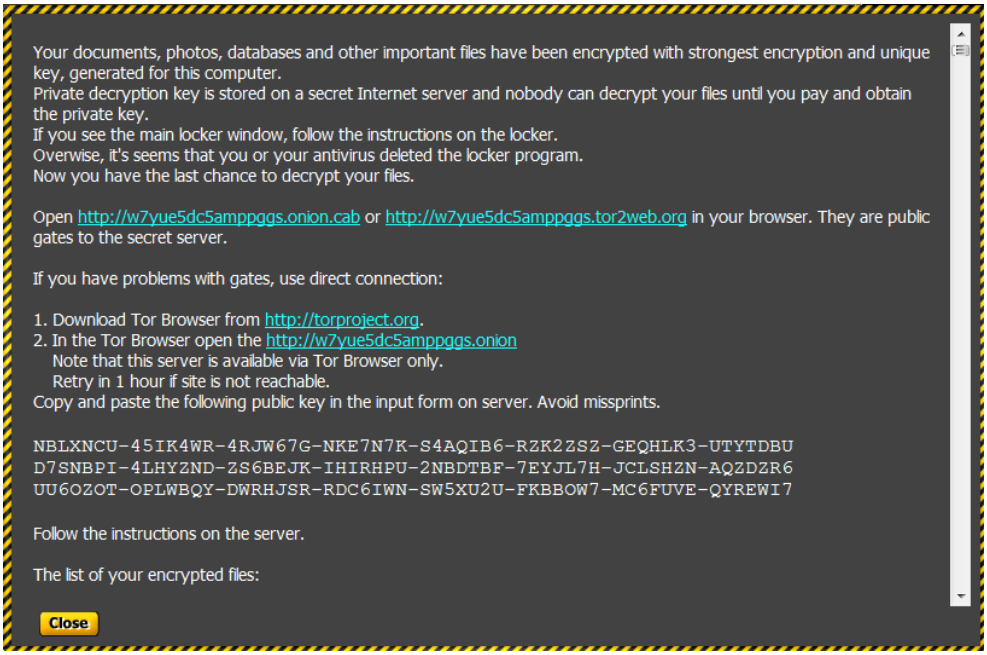
Screenshoty obrazovky z napadeného systému

Obrázek B.1: Pozadí plochy napadeného počítače



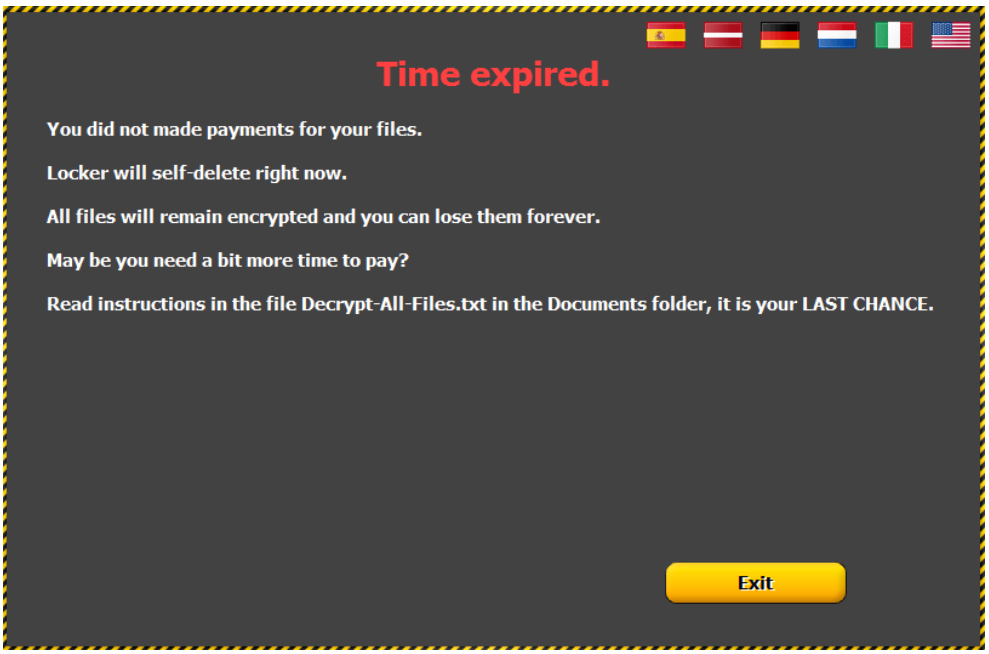
Obrázek B.2: Informace o platbě

Obrázek B.3: Test dešifrování – volba



Obrázek B.4: Veřejný klíč

Obrázek B.5: Test dešifrování – soubory



Obrázek B.6: Vypršení času

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD a hesla k archivům
- `src`
  - `Process monitor` ..... Záznamy z programu Process monitor
  - `dumpy` ..... Surové obsahy pamětí
  - `malware.idb` ..... Soubor pro program IDA
  - `other`
    - `Decrypt-All-Files.bmp` Obrazový soubor informující o zašifrování
    - `Decrypt-All-Files.txt` . Textový soubor informující o zašifrování
    - `otzcwtc.html` . HTML tabulka se seznamem zašifrovaných souborů
  - `sit`
    - `dns.pcapng` ..... Zachycený DNS provoz
    - `tor.pcapng` ..... Zachycený Tor provoz
  - `vzorky malwaru`
    - `1_vzorek.zip` ..... viz. kapitola 5]
    - `2_vzorek.zip` ..... viz. kapitola 6
    - `rekonstruovany_malware.zip` ..... viz. kapitola 6
  - `thesis`
    - `thesis.tex` ..... zdrojová forma práce ve formátu LaTeX
    - `picture` ..... grafické podklady pro text práce
- `text` ..... text práce
  - `DP_Řečínský_Jan_2016.pdf` ..... text práce ve formátu PDF