VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

NASAZENÍ A VYLEPŠENÍ NÁSTROJE PRO ZACHYTÁVÁNÍ RDP ÚTOKŮ

DEPLOYMENT AND ENHANCEMENT OF TOOL FOR CAPTURING RDP ATTACKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DANIEL SNÁŠEL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. LUKÁŠ ZOBAL**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Snášel Daniel**
Program: Informační technologie
Název: **Nasazení a vylepšení nástroje pro zachytávání RDP útoků**
Deployment and Enhancement of Tool for Capturing RDP Attacks
Kategorie: Bezpečnost

Zadání:

1. Seznamte se s protokolem Remote Desktop Protocol (RDP) a jeho existujícími problémy z hlediska kyberbezpečnosti.
2. Seznamte se s problematikou honeypotů se zaměřením na RDP honeypoty.
3. Po konzultaci s vedoucím nasaďte vybraný nástroj na veřejně dostupný server, vyhodnoťte jeho schopnosti a navrhněte vylepšení, která povedou k efektivnějšímu sběru dat.
4. Návrh implementujte a otestujte.
5. Vyhodnoťte získaná data z nasazeného honeypotu a zhodnoťte dopad implementovaných vylepšení.

Literatura:

- Nawrocki, Marcin, et al. "A Survey on Honeypot Software and Data Analysis." arXiv preprint arXiv:1608.06249 (2016).
- https://www.trustedsec.com/2019/01/adventures-of-an-rdp-honeypot-part-one-rdp-security/
- https://nakedsecurity.sophos.com/2019/07/17/rdp-exposed-the-wolves-already-at-your-door/
- https://www.gosecure.net/blog/2018/12/19/rdp-man-in-the-middle-smile-youre-on-camera

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, částečně bod 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Zobal Lukáš, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 22. října 2019

Abstrakt

Honeypoty jsou široce používané pro výzkum počítačové bezpečnosti. Cílem honeypotů je pomáhat bezpečnostním výzkumníkům získávat cenné informace o útočnících na síti. Tato práce se zabývá návrhem a vylepšením stávajícího honeypotu PyRDP. Nejprve práce popisuje honeypoty a představuje jejich základní myšlenky. Dále popisuje problematiku vzdálené plochy a její zranitelnosti. V rámci této práce jsou navrženy a implementovány vylepšení existujícího vysoce interaktivního honeypotu PyRDP. Tento nástroj je řádně otestován a jsou analyzována data získaná z jeho nasazení.

Abstract

Honeypots are widely used in computer security research. Their task consists of assisting security researchers in gaining valuable information about network attackers. This thesis deals with the design and improvement of the existing PyRDP honeypot. First, honeypots in general are described along with the basic concept of those. Then, the issues of the remote desktop and its vulnerability are described. Finally, the improvements of already existing highly interactive PyRDP honeypot are proposed and implemented. This tool has been properly tested and the analyzed data were obtained from its deployment.

Klíčová slova

honeypot, rdp, protokol vzdálené plochy, pyrdp, rdpy, honeypot s vysokou mírou interakce

Keywords

honeypot, rdp, remote desktop protocol, pyrdp, rdpy, high-interaction honeypot

Citace

SNÁŠEL, Daniel. *Nasazení a vylepšení nástroje pro zachytávání RDP útoků*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Lukáš Zobal

Nasazení a vylepšení nástroje pro zachytávání RDP útoků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Lukáše Zobala. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Daniel Snášel
24. května 2020

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Lukášovi Zobalovi za pomoc při výběru tématu, cenné rady, poskytnutí materiálů a odborné vedení bakalářské práce.

Obsah

Kapitola 1

Úvod

Od doby, kdy byly poprvé počítače vzájemně propojeny přes síť, uběhlo již mnoho let. V průběhu několika let se počet vzájemně propojených počítačů v celosvětové síti Internet zvýšil několikanásobně. Se zvyšujícím se počtem počítačů se vyvinula i hrozba ve formě sledování a odposlouchávání uživatele. Vynalézavost útočníků v tomto odvětví nebere mezí, a proto je pro pracovníky v oblasti síťové bezpečnosti složité udržet krok s novými metodami útoků. Bohužel počet a sofistikovanost útoků stále roste a pasivní obrana přestává být účinná. Je zapotřebí použít jinou technologii na zastavení útočníka. Relativně novým způsobem ochrany jsou honeypoty. Toto řešení je agresivnější a efektivnější než klasická obrana proti útokům. Pomocí honeypotů lze zastavit útočníka ještě před vstupem do našeho systému.

Termín honeypot lze do českého jazyka doslovně přeložit jako „hrnec medu“. Tato varianta překladu je plně dostačující pro základní vysvětlení technologie. Stejně jako je hrnec medu položený na volně přístupném místě potravou pro hmyz, který se kolem shromažďuje proto, aby med získal, tak veřejně umístěný honeypot v síti Internet, který připomíná jednoduchou kořist, je potravou pro útočníky snažící se prolomit ochranu počítače a dostat se dovnitř. Útočník nesmí vědět, že počítač, na který útočí je honeypot.

Cílem práce je seznámit čtenáře blíže s problematikou vzdálené plochy, honeypotů, jejich dělení a hlavními rozdíly. Dále pak vymyslet a navrhnout vlastní architekturu pro nasazení honeypotu protokolu vzdálené plochy, který bude nabízet co možná největší míru interakce. Hlavním požadavkem je logování a sledování útočníka po celou dobu jeho útoku.

Text práce je rozdělen do tří částí. První část je teoretická, je v ní vysvětlen a definován termín honeypot v kapitole 2 a jsou zde uvedeny nejběžnější způsoby jejich dělení a použití. Hlavní důraz se zde klade na rozdělení dle míry interakce s útočníkem. Dále je pak detailně rozebrána problematika vzdálené plochy v kapitole 3. Zde se klade největší důraz na bezpečností chyby protokolu. Následuje popis vybraného honeypotu s názvem PyRDP [1]. Tento honeypot v kapitole 4 dovoluje útočníkovi přihlásit se pomocí protokolu vzdálené plochy na koncové zařízení. Při spojení útočníka s koncovým zařízením přes honeypot je spojení nahráváno. Druhá část se zaměřuje na praktické nasazení a vylepšení celkové architektury honeypotu pro sběr dat. Tento princip je popsán v kapitolo4.

V další části je detailně popsaná problematika implementace honeypotu 5. Přidáním druhé cesty do architektury honeypotů je možné zachytit daleko více zkušených útočníků, kterým nejde jen o prolomení hesla pomocí útoku hrubou silou. Zkušení útočníci dokáží využít získaný počítač pro svoje další plány a zanechat na tomto počítači zadní vrátka pro další připojení. Nejčastěji je ale získaný počítač používán na skenování dalších počítačů ve stejné nebo celosvětové síti, případně na jejich napadení. Pokud se bude jednat jen

o nezkušeného útočníka, tak ten se většinou pokouší jen stáhnout veškerý cenný obsah z disku a následně poškodit počítač tak, aby se k němu původní majitel již nikdy nedostal. Této problematice se věnuje kapitola 6 nazvaná Analýza dat.

Poslední kapitola 7 této části práce pojednává o testování jednotlivých skriptů architektury a monitorování honeypotu. Pomocí testování lze odhalit chyby, které by mohl útočník využít k dalším útokům.

Kapitola 2

Problematika honeypotů

Kapitola je věnována obecnému konceptu honeypotů, míře užitečnosti v počítačových sítích a zároveň celkovému významu vzhledem k bezpečnosti. Kapitola obsahuje zmínku o základním rozdělení honeypotů do jednotlivých sekcí podle jejich vlastností [4].

2.1 Definice honeypotů

Pod pojmem honeypot si každý může představit něco odlišného. Pro jednoho může představovat komplexní záznamník všech útočníků a jejich kroků v případě napadení počítače, jiný v tom může vidět účinný prvek pro budoucí ochranu ostatních počítačů v rámci celosvětové sítě Internet. Díky vývoji honeypotů a jejich velké škále tak není jednoduché přesně definovat formální pojem honeypot. Lance Spitzner definuje honeypot následovně [16].

„Honeypot je bezpečnostní prvek, jehož hodnota spočívá v tom být zkoušen, napaden či zneužit."

Tento prvek bezpečnosti může být například služba, aplikace, systém nebo skupina systémů. V podstatě cokoliv, co dokáže zaznamenat data útočníka, popřípadě celý jeho útok. Jelikož honeypot sám neobsahuje žádná data ke zneužití, ani nemá žádnou produkční hodnotu, stroj ani osoba nemá důvod s tímto počítačem komunikovat. Hlavním předpokladem příchozí komunikace je, že jakákoliv přistupující entita k tomuto prvku v síti je podezřelá. Aby honeypot splňoval svoji funkci, musí být pro útočníka zajímavý. Zároveň by nemělo být na venek rozpoznatelné, že jde o past, mělo by to působit jako slabě zabezpečený počítač. Honeypot by měl napodobovat produkční počítač co nejvěrněji, aby přilákal co nejvíce útočníků [15].

2.2 Využití honeypotů

Honeypot není omezen na konkrétní účel, naopak se jedná o velmi flexibilní nástroj. Proto je i jeho definice viz sekce 2.1 poměrně obecná. Jsou v ní ovšem popsány hlavní přínosy honeypotů.

Jelikož cíle útočníka mohou být různé, stejně tak i způsoby provedení útoků se velmi liší. Napadení může proběhnout prostřednictvím škodlivého softwaru, jako jsou nejrůznější druhy škodlivých kódů, nebo přímou interakcí útočníka, jenž může být amatér nebo zkušený profesionál. Řada škodlivého softwaru má za úkol provedení prostého mapování sítě a na-

lezení spuštěných služeb k budoucímu útoku. Při napadení se může jednat o útok hrubou silou se záměrem proniknout na účet správce služby či systému pomocí uhodnutí přihlašovacích údajů. Tento typ útoků je velmi pomalý a na síti lehce rozpoznatelný, proto se zkušení útočníci zaměřují na jiný typ útoků, a to na využití slabin konkrétní verze určité služby [12].

Reakce na útoky

Na útoky těchto typů lze reagovat mnoha odlišnými způsoby. Nekalá aktivita může být rozpoznána a následně zablokována. Pokud je cílem získat všechny kroky útočníka o provedeném útoku, jeho aktivita bude povolena a sledována, případně se upraví komunikace mezi útočníkem a honeypotem pro lepší sběr dat. Komunikaci lze upravit zpomalením provozu tak, aby útočník nic nepoznal, nebo podvržením certifikátu pro lepší sledování. Nezávisle na typu útoků, jež byly rozebrány, musí být veškerý síťový provoz mezi honeypotem a útočníkem monitorován, aby s nasbíranými daty bylo možné dále pracovat a provádět nad nimi analýzu [17].

Honeypot jako bezpečnostní prvek

Honeypot může sloužit jako vrstva ochrany proti útokům, především díky odvedení pozornosti od reálných systémů na síti. Tím, že jsou útočníkovi poskytnuty zajímavější cíle, kterým dá přednost, protože obsahují mnoho slabin, se snižuje šance napadení produkčního zdroje. Tyto zranitelnosti se ponechávají viditelně na honeypotu a nasměrovávají tak útočníka přímo do honeypotů. Honeypot je v ideálním případě oddělen od produkční sítě a útočník se díky správnému nastavení a přesměrování nemůže dostat k cenným informacím, a to i v případě rozpoznání, že se jedná o honeypot. V takovém případě si honeypot může zachovat svůj stav v rámci ochrany sítě. Za předpokladu správného nastavení honeypotu je pro útočníka lepší najít si za cíl jinou organizaci, než riskovat další honeypot v síti. Pokud by byl honeypot špatně nastaven, tak může být kompromitován a útočník se pomocí něj může dostat do vnitřní sítě organizace.

Honeypot jako bezpečnostní prvek je pouze okrajové využití, protože tuto ochranu útočník může velmi snadno odhalit a jednoduše se jí vyhnout. Většinou se honeypot vyžívá pro sběr dat od útočníka.

Honeypot jako detektor útoků

Jednou z funkcí, kterou honeypot zvládá velmi dobře, je detekce útoků. Pro rozpoznání útoků na síti je nutné rozdělit podezřelý provoz od regulérního. Výhodou, kterou honeypoty poskytují v tomto ohledu, je téměř absolutní jistota, že jakákoliv interakce s nimi je podezřelá. Pro tuto funkcionalitu je nejlepší umístění honeypotu do sítě ke strojům, u nichž lze předpokládat zvýšené riziko napadení. Při skenování sítě, kdy útočník zjišťuje otevřené porty a jaké služby na nich běží, honeypot včas upozorní před nekalým jednáním útočníka. Honeypoty na útočníkovu komunikaci nereagují a pouze pasivně monitorují síťový provoz. Těmto typům honeypotů se přezdívá honeypoty bez interakce nebo honeypoty s nulovou interakcí.

Získávání dat od útočníka

Analýza škodlivého kódu hraje důležitou roli v bezpečnosti informačních technologií. Honeypot může fungovat jako zachytávač škodlivého kódu, který se následně může analyzovat, aby bylo možné tento kód v budoucnosti neutralizovat.

Nejlepším zdrojem informací o útocích jsou samotní útočníci, kteří nás mohou naučit nejvíce o svých postupech. Prvotřídním způsobem, jak se dozvědět o jejich technikách, je sledovat je přímo při akci. K tomuto účelu slouží nejlépe honeypoty s vysokou mírou interakce. Ty umožňují sledovat útok krok za krokem, příkaz za příkazem a jakým způsobem byl útok proveden.

Sledování útočníků

Pomocí těchto technik lze spatřit nové techniky útoků a nové zranitelnosti. Jelikož neznámé zranitelnosti se jen těžko brání, je jejich objevení velice cenné. Díky honeypotu vzniká možnost vysledování zranitelností systému, o kterých jsme dosud neměli ani tušení. Některé zranitelnosti mohou být i na produkčním serveru a je třeba ihned po jejich objevení zabezpečit produkční server tak, aby nebylo možné se při použití stejné zranitelnosti dostat k citlivým datům.

Z charakteristiky a přínosů honeypotů vyplývá, že se nejedná o řešení celkové síťové bezpečnosti. Jejich hodnota spočívá především v poskytnutí cenných informací o stavu sítě a technikách útočníka. Tyto poznatky lze aplikovat ke zlepšení současných nedostatků v síti.

2.3 Rozdělení honeypotů

Tato podkapitola je zaměřená na detailní rozdělení honeypotů podle jejich vlastností. Honeypoty je možné dělit podle několika kritérií, například dle interakce, protokolu a umístění. Jednotlivé dělení je popsáno níže [11].

2.3.1 Dle interakce

Prvním z mnoha hledisek je dělení dle interakce. Toto kritérium rozděluje honeypoty do tří skupin, a to na honeypoty s nízkou, střední a vysokou mírou interakce z pohledu útočníka. Čím větší interakci honeypot přináší, tím je na první pohled méně rozeznatelný od reálných systémů a umožňuje podrobnější sběr dat.

Nízká míra interakce Honeypoty s nízkou mírou interakce zpravidla napodobují jen určité služby, aplikace nebo jiné části systému. Z pohledu útočníka honeypot s nízkou mírou interakce reaguje jen na dotazy, které tvůrce honeypotu implementoval. Stejně je tomu tak v případě zranitelností, které zde autor zanechal schválně. Vše tedy funguje v omezené míře oproti skutečné službě, případně systému. Nevýhody tohoto druhu honeypotu jsou, že zkušený útočník může snáze rozeznat past od regulérní aplikace či systému.

U honeypotů s nízkou mírou interakce je potencionální sběr dat o útočníkovi značně omezen. Tato omezení snižují hodnotu získaných informací o technikách útočníka oproti honeypotům s vyšší mírou interakce. Data sbíraná pomocí těchto honeypotů jsou především kvantitativního charakteru.

Hlavním přínosem tohoto druhu honeypotů je sběr škodlivého softwaru a hesel. Mezi zástupce s tímto účelem patří Amun[1] či Heralding[2].

Výhodou honeypotů s nízkou mírou interakce je jejich snadné, rychlé nasazení a jejich bezúdržbovost. Hardwarové nároky na provoz honeypotů jsou poměrně nízké, zejména ve srovnání s honeypoty s vysokou mírou interakce. Kromě jednoduchého nasazení a správy mají honeypoty s nízkou mírou interakce ještě jednu hlavní výhodu. Pro tento druh honeypotů je nepravděpodobné, aby se jich útočník zmocnil, a proto při jejich napadení neohrožuje okolní síť.

Při použití honeynetu neboli sítě honeypotů může být honeypot s nízkou mírou interakce využit pro přesměrování komunikace určitých portů na jiný honeypot s vysokou mírou interakce. Tento honeypot pak může plně emulovat určitou službu.

Střední míra interakce Honeypoty se střední mírou interakce jsou kompromisem mezi honeypoty s malou a vysokou mírou interakce. Tento honeypot se snaží co nejvíce emulovat danou službu. Některé funkcionality ale nemusí být implementovány. Z pohledu útočníka honeypot se střední mírou interakce reaguje na všechny druhy dotazů, pokud je autor implementoval. Stejně tak v případě Zranitelnosti konkrétní služby. Nevýhodou tohoto druhu honeypotu je, že zkušený útočník může rozeznat past, jestliže je detailně obeznámen s protokolem a s danou službou běžící nad tímto portem.

U honeypotů se střední mírou interakce je potenciální sběr dat o útočníkovi rozsáhlejší než v případě honeypotů s nízkou mírou interakce, ale zásadně menší než v případě honeypotů s vysokou mírou interakce. Data sbíraná tímto typem honeypotů jsou především kvantitativního charakteru a přidělená útočníkovi dle IP adresy.

Hlavním přínosem těchto honeypotů je sběr škodlivého kódu, hesel a souborů. Mezi zástupce s tímto účelem patří například Cowrie [13]. Obecně honeypoty se střední mírou interakce umožňují detekovat nové způsoby útoků díky ukládání velkého množství dat.

Výhodou honeypotů se střední mírou interakce je jejich snadné a rychlé nasazení do sítě. Lze je velmi rychle nakonfigurovat tak, aby jejich zneužití nenapáchalo škody v síti.

Nevýhodou těchto honeypotů je nutnost údržby a vůči honeypotu s nízkou mírou interakce jsou i náročnější na hardwarové požadavky. Při špatném nakonfigurování honeypotu se jej může útočník zmocnit a napáchat tak škody v síti.

Vysoká míra interakce V případě honeypotů s vysokou mírou interakce je plně napodoben skutečný systém nebo služba. V případě napodobení reálného systému se může jednat například o počítač, směrovač či jiné prvky v síti, které ale s síti nemají žádnou skutečnou funkci, a tudíž by s nimi neměl žádný uživatel komunikovat.

Rozdíl mezi předchozími typy honeypotů je ten, že honeypot s vysokou mírou interakce napodobuje, nebo je, skutečný operační systém, včetně aplikací a služeb, pouze s drobnými změnami umožňujícími jejich monitorování.

Hlavními funkcemi honeypotu s vysokou mírou interakce jsou možnosti podrobného sledování chování útočníka, objevování nových zranitelností a sběr doposud neznámého škodlivého softwaru.

Výhodou tohoto typu honeypotů je jejich věrnost. Ani zkušený útočník si na první pohled nemusí uvědomit, že byl lapen do pasti. Pokud se jedná o skutečný systém, je poté možná nejvyšší možná míra interakce s útočníkem. Ovšem i u těchto honeypotů jsme do

[1] https://github.com/zeroq/amun
[2] https://github.com/johnnykv/heralding

jisté míry limitovaní verzemi aplikací nebo služeb. Špatně nastavený honeypot s vysokou mírou interakce může být rozpoznán například podle výchozích přihlašovacích údajů, pokud zde nejsou ponechány z dobrého důvodu. Mezi další identifikátory patří například minimum běžících procesů, žádná data nebo naopak příliš vhodná data.

U tohoto druhu honeypotu by mělo být navrženo důsledné sledování, a to kvůli velkému množství nasbíraných dat o útočnících a jejich metodách. Podrobná následná analýza může zabrat až několik dní. S tím souvisí i jedna z hlavních nevýhod honeypotů s vysokou mírou interakce, a to jsou vysoké nároky na správu. Dalšími nevýhodami jsou obtížnější konfigurace a nasazení do sítě, než tomu bylo u předchozích typů.

U honeypotů s vysokou mírou interakce je potřeba vzít v úvahu jejich hardwarové nároky, jelikož nasazení skutečného systému je přirozeně náročnější na výkon, než pouhá simulace jeho části.

Největší nevýhodou je vyšší riziko ohrožení okolní sítě. Pokud se útočník zmocní tohoto typu honeypotů, má následně možnost instalovat nový software a provádět změny v operačním systému, čímž může ohrozit všechny počítače v dané síti. Jelikož se útočník může pokusit po sobě zamést stopy a smazat logy obsahující záznamy o jeho aktivitě, je vhodné tato data přeposílat na vybrané úložiště. Tato problematika bude více popsána později.

2.3.2 Dle protokolů

Druhé kritérium dělení honeypotů je podle protokolů. Tímto kritériem rozdělujeme honeypoty do mnoha skupin dle jejich portů a rozhraní, na kterých běží simulované služby nebo celé systémy.

Každá služba má přidělené porty, na kterých může komunikovat. Na těchto portech se většinou očekávají pouze služby, které by na nich měly běžet. Příkladem takové služby a portu je HTTP na portu 80, případně HTTPS na portu 443. Z hlediska této práce je nejdůležitější port 3389 a služba Remote Desktop Protocol (dále už jen RDP). Proto bude tento protokol později podrobně popsán.

U každého protokolu se nasazuje honeypot kvůli bezpečnosti, ale nemusí vždy umět všechny vlastnosti, jež definují honeypot. U některých protokolů je třeba zkoumat jen dostačující kvalita hesel a poslaný škodlivý kód. Jiné protokoly vyžadují honeypoty s vysokou mírou interakce, aby bylo možné zkoumat útočníkovo chování krok za krokem.

2.3.3 Dle umístění v počítačové síti

Tradiční honeypoty se nacházejí na straně serveru, existují ale i honeypoty které se nacházejí na straně koncové. Tato podsekce bude zaměřena především na serverové honeypoty.

Na klientské straně Klientské honeypoty, jak už název napovídá, připomínají samotného klienta. Tento druh honeypotu se aktivně přihlašuje ke vzdáleným nebo lokálním službám a následně kontroluje, zda nebyla provedena změna systému, popřípadě jestli nedošlo k jeho napadení. Cílem honeypotu je detekovat útoky na klientské aplikace, zpravidla jde o detekci škodlivého chování či obsah nějakého serveru. Honeypot tedy sám aktivně vyhledává služby, ke kterým se následně připojí. Obzvláště pro klientské honeypoty platí, že musí být umístěné ve zvláštní síti a striktně monitorovány, protože je zde velká pravděpodobnost jejich napadení.

Na serverové straně Honeypoty na serveru pasivně poslouchají na standardních portech u vybraných služeb a sledují jakékoliv spojení s útočníkem. Tyto honeypoty mají za cíl obvykle zjišťovat nové typy hrozeb a sbírání škodlivého softwaru. Příkladem pozorovací služby je například SSH, kdy honeypot sleduje uživatele, kteří se pokoušejí připojit ke stroji a získat přístup na některý z účtů. Serverové honeypoty mohou útočníka přitahovat pomocí lákavých jmen a uvítací zprávy od dané služby.

2.4 Detekce honeypotů

Ačkoliv se honeypot snaží o simulaci služby, aplikace, systému či jiných prostředků, není vždy dokonalý, a proto může zkušený útočník poznat, že jde o honeypot ještě dříve, než se vůbec o útok pokusí.

Způsob detekce je samozřejmě odlišný u honeypotu s nízkou a vysokou mírou interakce [12]. U honeypotu s nízkou mírou interakce, kde je emulovaná jen určitá služba, nikoliv celý systém, se útočník zaměřuje na počet otevřených portů, na verzi dané služby a na výpis uvítací zprávy. Pokud útočník zjistí nějaké nesrovnalosti, většinou se ani nepokusí o nějaký rafinovaný útok a raději začne prozkoumávat jiný cíl.

Detekce u honeypotů s vysokou mírou interakce se zaměřuje na systém jako celek, zkoumá se zde síťová komunikace, počet otevřených portů. Stejně jako v případě při detekci honeypotu s nízkou mírou interakce. U tohoto typu k tomu ještě přibývá detekce počtu běžících programů a data která se na koncovém zařízení nacházejí. Pokud se na koncovém zařízení nenacházejí data, útočník většinou ihned opouští stroj [9]. To samé platí v případě nalezení až příliš dokonalých dat.

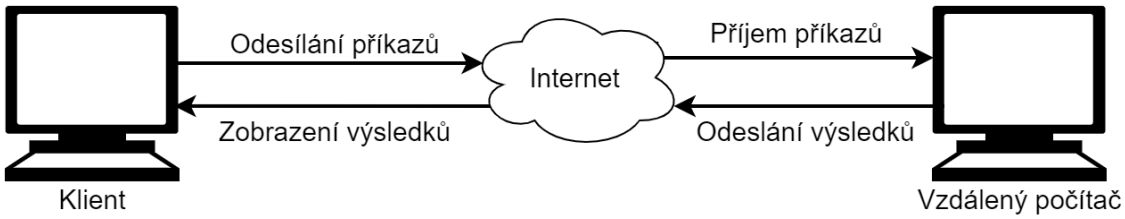
Nástroj vhodný pro detekci honeypotu je například Nmap[3]. Tento nástroj dokáže o koncovém zařízení zjistit: kolik má otevřených portů, které to jsou, jaký operační systém používá a další velmi užitečné informace, které zkušenému útočníkovi usnadňují jak detekci, tak napadení koncového zařízení.

[3] https://nmap.org/

Kapitola 3

Protokol vzdálené plochy

Protokol vzdálené plochy umožňuje obsluhu vzdáleného počítače tak, jako kdyby uživatel seděl přímo u daného počítače, přičemž může být v geograficky odlišné poloze. Pomocí tohoto protokolu se může připojit na daný počítač pomocí celosvětové sítě Internet. Celá kapitola je věnována právě protokolu vzdálené plochy, protože honeypot který je hlavní dominantou této práce, sleduje chování útočníků nad tímto protokolem.



Obrázek 3.1: Schéma protokolu RDP

3.1 Popis protokolu vzdálené plochy

Anglicky Remote Desktop Protocol [3] (dále už jen RDP) je síťový protokol, který umožňuje uživateli ovládat vzdálený počítač s grafickým rozhraním a s operačním systémem Windows XP a novější nebo jiný operační systém s podporou RDP. Připojení funguje na principu klient-server, kdy je uživatel v roli klienta a pomocí jednoduchého grafického rozhraní se připojí na server, který následně můžu plnohodnotně ovládat tak, jako kdyby byl fyzicky u daného počítače. RDP server implicitně naslouchá na protokolu transportní vrstvy s číslem portu 3389 [3], viz obrázek 3.1. Protokol vzdálené plochy je šifrovaný. Při navazování spojení tudíž dojde k výměně certifikátů. Od novějších verzí protokolu je potřeba znát přihlašovací údaje na vzdálený počítač ještě před začátkem sezení. Uživatel je nejprve vyzván k zadání přihlašovacích údajů. Následně si vymění certifikát se vzdáleným počítačem a zkontroluje se správnost přihlašovacích údajů. Ve starších verzích protokolu si nejprve uživatel vyměnil certifikát se vzdáleným počítačem a následně se na něj připojil. Po úspěšném připojení byl uživatel vyzván k zadání přihlašovacích údajů. Postup byl změněn kvůli minimalizování náročnosti na výkon počítače [14].

3.2 Bezpečnost a chyby protokolu

Hlavní nevýhodou protokolu vzdálené plochy je možnost útoků hrubou silou. Proti tomuto útoku není protokol nijak zabezpečen. Jediným způsobem, jak se chránit, je používat dostatečně silné heslo, které není prolomitelné v reálném čase.

Tento síťový protokol využívá certifikáty, které slouží k ověření komunikace mezi klientem a serverem. Zároveň slouží k šifrování komunikace. Pokud se útočník postaví do síťové komunikace mezi klienta a server, tak při prvním připojení klienta na server útočník přesměruje komunikaci na server, aby uživatel nic nepoznal, a získá tak od uživatele certifikát. Pomocí certifikátu od uživatele může rozšifrovat komunikaci a znovu ji zašifrovat, aby ji mohl šifrovaně odeslat na server. V tomto případě útočník vidí veškerou komunikaci mezi klientským programem a cílovým počítačem. Tento typ útoků se nazývá Man-In-The-Middle [2] (dále už jen MITM). Na stejném principu pracuje již zmíněný PyRDP, který bude detailně popsán v následující kapitole [6].

Útočník může zaslat na server nejen přihlašovací údaje při autorizaci, ale i škodlivý kód, který mu následně dovolí ovládnout vzdálené počítače [10]. Jedna z takových zranitelností se nazývá BlueKeep[1]. Tento škodlivý kód umožňuje útočníkovi připojit se na server bez nutnosti autorizace. Útočník ihned po připojení pomocí této zranitelnosti získává oprávnění administrátora a celý počítač je tak pod jeho kontrolou [5].

3.3 RDP honeypoty

Kvůli častým útokům na RDP byly vytvořeny honeypoty s nízkou a vysokou mírou interakce. Honeypoty s nízkou mírou interakce mají za cíl odchytávat útoky hrubou silou a tak zjišťovat, jestli je heslo na serveru dostačující a jestli se nenachází v některém slovníku, které se používají na podobné typy útoků. Další z jejich hlavních cílů je odlákání útočníka od regulérního serveru. Případně ještě můžou odchytávat škodlivý kód, jenž může útočník poslat. Příkladem takového honeypotu je Heralding [18] a jeho modul RDP.

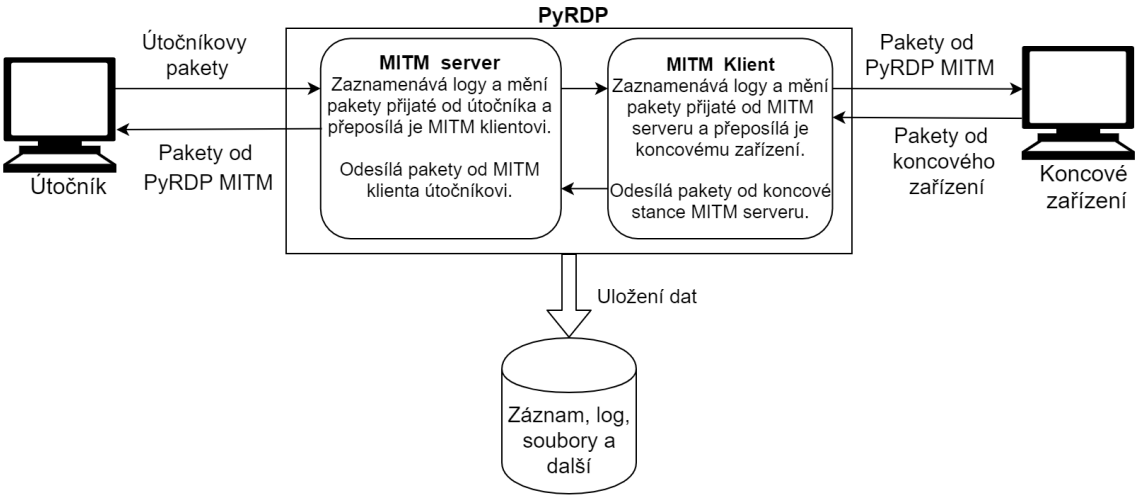
Honeypoty s vysokou mírou interakce mají za cíl to samé jako honeypoty s nízkou mírou interakce rozšířené o pozorování útočníka po připojení se na server a jeho dalšího konání. Příkladem takového honeypotu je PyRDP, který bude detailně popsán v další kapitole. Dalším zástupcem je například RDPY. Většina honeypotů s vysokou mírou interakce nad protokolem vzdálené plochy funguje na principu MITM. RDP je šifrovaný protokol, a proto není možné číst data bez znalosti certifikátů. Certifikáty si mezi sebou vymění útočník a koncová stanice. Řešením je tedy MITM, který stojí uprostřed síťové komunikace, a má tedy přístup k certifikátům. Výměna certifikátů a jejich podvržení je popsaná v další kapitole 4. Detailnější popis honeypotů s vysokou mírou interakce viz sekce rozdělení honeypotu 2.3.

[1] https://blog.avast.com/cs/what-is-bluekeep

Kapitola 4

PyRDP honeypot

PyRDP je honeypot s vysokou mírou interakce využívající techniky MITM, viz obrázek 4.1. Celková architektura pro připojení útočníka ke koncovému zařízení vypadá následovně. Útočník zaútočí na server s PyRDP (dále už jen server) s domněnkou, že se jedná o cílový počítač. Server začne přeposílat data od útočníka na koncové zařízení a od koncového zařízení zase zpět k útočníkovi. Při začátku komunikace si útočník vymění se serverem certifikáty a začne komunikace. O tomto kroku by neměl mít útočník tušení. V případě zjištění, že se jedná o past, by se mohl odpojit a rozeslat tuto zprávu dalšímu útočníkovi a server už by nemohl sbírat reálná data. Detekce honeypotů byla rozebrána dříve.



Obrázek 4.1: Schéma PyRDP MITM

4.1 Popis nástroje PyRDP

Tento honeypot umožňuje podvrhnout certifikát útočníkovi, aby bylo možné odposlouchávat komunikaci. Dále uchovává přihlašovací údaje použité při přihlašování a kopie souboru, jenž byly odeslány přes síť. Dokáže dále spustit programy v PowerShellu po přihlášení útočníka. Ovšem jeho hlavní výhodou je ukládání a zpětné přehrání záznamů, které slouží k pochopení útočníkovy techniky nejlépe.

Celkově se tento nástroj skládá ze tří částí. První je samotný honeypot, který se stará o komunikaci mezi útočníkem a serverem. Ukládá přenesené soubory, obsah schránky, veškerá data zadaná pomocí klávesnice a nahrané záznamy. Druhý je přehrávač záznamů, který dovoluje přehrát záznam, jenž byl nahrán při pokusu útočníka o přihlášení, případně při jeho další aktivitě. Záznam v sobě uchovává grafická data, stisknuté klávesy a textový obsah schránky. Poslední částí je skript, který umí vytvořit falešný certifikát za pomocí připojení se na koncové zařízení a zkopírování daného certifikátu.

Tento certifikát lze následně použít při spuštění honeypotu. Takto vytvořený certifikát obsahuje chyby, které můžou útočníka upozornit, že se jedná o honeypot. Příkladem takové chyby je, že certifikát není ověřen certifikační autoritou nebo certifikát obsahuje jiné jméno počítače.

4.2 Nasazení vybraného honeypotu

K nasazení honeypotu je zapotřebí mít nainstalovaný operační systém Ubuntu 18.04 a novější, nebo Windows 7 a novější. Dalším potřebným softwarovým vybavením je balíček Python 3.5 a novější. Další kroky budou zaměřené na operační systém Ubuntu. Nástroje PyRDP lze nainstalovat různými způsoby. První z těchto způsobů je Docker Image, druhý způsob je pomocí git repozitáře a následně postupovat podle připraveného manuálu[1].

Po úspěšné instalaci a spuštění virtuálního prostředí je zapotřebí získat certifikát, který bude následně podvržen útočníkovi. Pokud žádný certifikát není k dispozici, nástroj certifikát vygeneruje. Tento certifikát slouží pouze k ověření funkčnosti, nikoliv na reálný provoz. Pro případ získání falešného certifikátu nástroj PyRDP obsahuje program, pomocí kterého je možné naklonovat certifikát z koncového stroje. Tento certifikát obsahuje reálná data, ale jak již bylo zmíněno, obsahuje i mnoho chyb. Posledním a nejlepším způsobem je vytvoření reálného certifikátu na serveru, kde je umístěn i celý nástroj PyRDP.

Vytvoření reálného certifikátu K vytvoření reálného certifikátu na serveru s PyRDP je potřeba mít nainstalovaný webový server Apache2 s platnou doménou. Dále je potřeba program Certbot[2], ve kterém si uživatel může vytvořit certifikát. Tento certifikát následně může zkopírovat do složky s nástrojem PyRDP a používat ho jako certifikát, jenž bude podvržen útočníkovi pomocí honeypotu.

Vytvoření koncového zařízení Koncové zařízení může být fyzický nebo virtuální počítač. Velkou výhodou u tohoto typu honeypotu je, že nemusí být na stejném zařízení, ani ve stejné síti jako je nástroj PyRDP. Koncové zařízení je žádoucí naplnit vhodnými daty, aby přilákalo útočníka. Pokud je nasazený honeypot jen na zkušební dobu, je důležité, aby obsahoval jednoduché přihlašovací údaje. Tímto způsobem lze zachytit nejvíce amatérských útočníků. Koncové zařízení potřebuje určitý výkon, který je zde proto, aby útočník nemusel dlouho čekat, než se připojí a provede nějakou akci. Doporučuje se alespoň 4 GB operační paměti, 2 jádra na procesoru a alespoň 128 MB grafické karty.

[1] https://github.com/GoSecure/pyrdp
[2] https://certbot.eff.org/

4.3 Návrh vylepšení

Po nastudování nástroje PyRDP, který obsahuje spoustu užitečných věcí, a to od vytvoření falešného certifikátu až po nahrávání záznamu MITM PyRDP, jsem zjistil, že některé specifické věci neobsahuje. Neobsahuje, a to například odposlech koncové stanice, přenos souborů ve schránce a další věci, které jsem navrhnul jako vylepšení.

Odposlech koncové stanice

Nástroji PyRDP chybí podpora pro odposlech koncové stanice. Útočník se může připojit na koncové zařízení pomocí serveru, který obsahuje spuštěnou instanci PyRDP MITM, nebo se může připojit přímo na koncové zařízení. V tento okamžik, pokud se útočník připojí na koncové zařízení napřímo, tedy bez připojení se na server s instancí PyRDP MITM, je pro celou architekturu neviditelný. Rozšířením architektury o možnost odposlouchávat útočníka na koncové stanici je možné útočníka pozorovat i bez honeypotu.

Přidání možnosti odposlouchávat koncové zařízení přináší větší možnosti z pohledu sledování útočníka. Pokud ale zkušený útočník tento odposlech odhalí, může odposlouchávání využít ve svůj prospěch nebo záznam smazat. Proto je důležité bezpečné uložení záznamu na zabezpečené místo. K bezpečnému přenosu dat z koncového zařízení slouží TCP protokol. Tento protokol bude více popsán v kapitole 5.

Pro bezpečnější přenos dat mezi koncovým zařízením a bezpečným úložištěm je lepší přidat ještě jednu vrstvu do celkové architektury, která bude data kontrolovat a následně je přeposílat. Tímto se zabezpečí přenesená data v případě odhalení odposlechu.

Přenos souboru ve schránce

PyRDP MITM umožňuje přenášet vložený text ve schránce z útočníkova počítače na koncové zařízení. Tato vymoženost slouží útočníkovi například ke zkopírování webové odkazy a spuštění skriptu na koncovém zařízení. Bohužel nástroji chybí možnost kopírovat stejným způsobem soubory nebo aplikace. Vylepšení schránky pro přenos souboru ve schránce povede k lepší emulaci reálného systému. Při pokusu o přenesení aplikace či souboru ve schránce se uloží kopie souboru i na disk, na kterém se nachází nástroj PyRDP.

Podpora nejnovější verze RDP

Přidáním podpory pro novou verzi protokolu vzdálené plochy, který byl přidán do operačního systému Windows 10 v březnové aktualizaci v roce 2019, se útočníkovi otevřou další možnosti a emulovaný systém se stane více věrohodným.

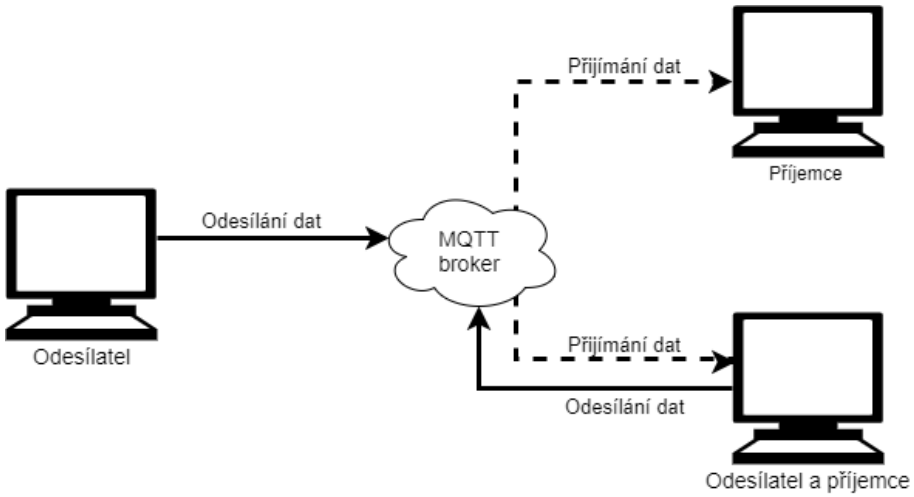
4.3.1 Protokol MQTT

Protokol Message Queuing Telemetry Transport[3] (dále už jen MQTT) slouží k publikování/odběru dat. Je to velmi jednoduchý a lehký protokol určený pro zasílání zpráv zařízením, u kterých se předpokládá malý výkon, například IoT zařízení, nebo pro zařízení nacházející se v sítí s vysokou latencí.

Protokol MQTT využívá připojení stroj-stroj (M2M). Proto dovoluje přenášet text i soubory mezi počítači za pomoci sítě. Vždy jedna strana vysílá a druhá strana přijímá komunikaci. Jeden počítač může přijímat nebo vysílat na více zařízení s různými IP adresami.

[3] http://mqtt.org/

Hlavní součástí protokolu jsou tři zařízení. První zařízení je odesílatel ("publisher"), ten odesílá data na zvolený port. Implicitní port pro nešifrovanou komunikaci je 1883 TCP a 8883 TCP slouží ke komunikace přes SSL. Druhé zařízení funguje jako příjemce ("subscriber"), jenž přijímá datový tok od odesílatele [8]. Na jednom fyzickém zařízení může být více odesílatelů či příjemců, popřípadě jejich kombinace. Posledním zařízením je propojovací prvek ("broker"), který se stará o přeposílání dat od odesílatele příjemci, viz obrázek 4.2.



Obrázek 4.2: Protokol MQTT

Návrat koncové stanice do původního stavu

Na základě výzkumu bylo zjištěno, že útočníci při svém útoku na koncové zařízení nehledají jen data. Někteří útočníci se snaží využít koncové zařízení pro své účely, jiní se snaží jen zařízení znepřístupnit dalším uživatelům, případně dalším útočníkům. Nejlehčím způsobem jak znepřístupnit koncové zařízení je změnit přihlašovací údaje. Kvůli změně přihlašovacích údajů se na server přihlásí daleko méně útočníků. Pokud se útočník rozhodne použít Ransomware[4], stává se zařízení nepoužitelné pro další sběr dat. Proto je zapotřebí mít připravený skript na obnovu koncového zařízení do původního stavu.

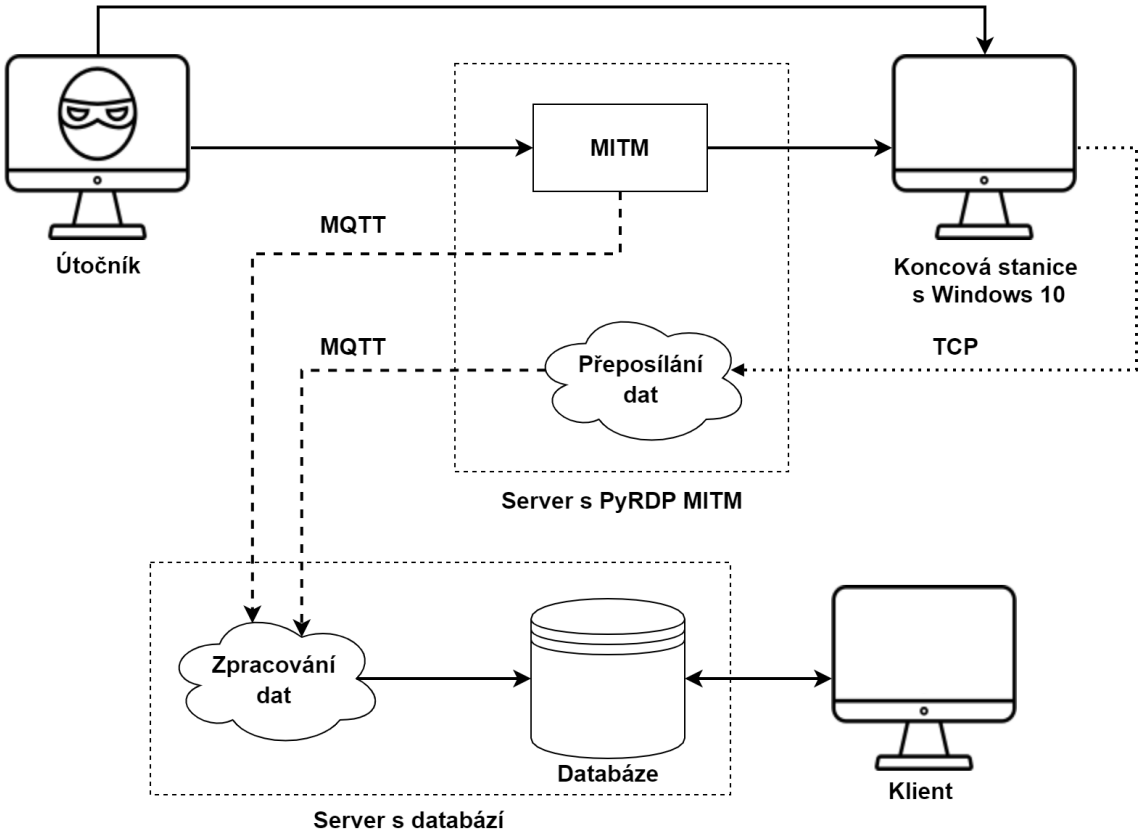
Pokud budeme využívat architekturu, která obsahuje virtualizační software a v ní koncové zařízení, můžeme použít například technologii tzv. snímků. Tato technologie dovolí uložit stav virtuálního stroje a kdykoliv je možné se k tomuto stavu vrátit. Následně stačí mít program, který provede tuto obnovu, a spouštěcí algoritmus. Například může docházet k obnovení stanice za nějaký časový úsek, nebo po odpojení útočníka.

4.4 Návrh celkové architektury

Celková architektura obsahuje útočníka, který se snaží připojit ke koncovému zařízení. Následně obsahuje server se spuštěnou instancí PyRDP MITM, jenž odesílá svoje záznamy pomocí MQTT přímo do programu na zpracování dat. Dalším prvkem je bezpečná databáze na uložení všech potřebných dat pro další analýzu. Předposlední částí je počítač, který je v této práci označován jako koncové zařízení. Obsahuje program TCPDump[5]. Ten slouží

[4] https://www.avast.com/cs-cz/c-ransomware
[5] https://www.tcpdump.org/

k získání škodlivého kódu a IP adresy útočníka, dále je zde k nalezení skript pro čtení znaků z klávesnice a ze schránky útočníka. Všechna data následně odešle pomocí protokolu TCP na server s PyRDP MITM, kde je připraven skript, jehož hlavním úkolem je přeposlat přijatá data na databázový server pomocí MQTT. Koncové zařízení obsahuje vhodná data, která mají za úkol nalákat a oklamat útočníka. Celá architektura je vyobrazena níže na obrázku 4.3.



Obrázek 4.3: Celková architektura

Celková architektura obsahuje dva způsoby připojení na koncové zařízení. První způsob obsahuje instanci PyRDP MITM, jenž zaznamenává všechny útočníkovy kroky a pomocí protokolu MQTT přeposílá záznamy programu, který je ukládá do databáze. Druhým způsobem je připojení přímo na koncovou stanici obsahující software na zaznamenávání útočníkova počínání a tato data jsou okamžitě přeposlána na jiný počítač pomocí protokolu TCP, aby je útočník nemohl za žádných podmínek vymazat. Data ukládaná do databáze obsahují záznam natočeného útočníka, všechny stisknuté klávesy od doby připojení, textový obsah schránky či soubory ve schránce, přenesené soubory, IP adresu útočníka nebo časový údaj o době připojení na koncovou stanici. Uživatel, který bude provádět analýzu dat a následný výzkum, už přistupuje jen do databáze, v níž se nacházejí všechna sesbíraná data.

Zobrazování dat

Součástí architektury jsou uložená data v databázi a jejich následné prohlížení klientem. V databázi budou uložena data o jednotlivých útocích. Každé sezení bude mít v databázi jednoznačný identifikátor. Celkový záznam v tabulce bude obsahovat následující:

- Jednoznačný identifikátor sezení,
- čas připojení,
- časový údaj o délce připojení,
- log z PyRDP MITM,
- záznam sezení,
- přenesené soubory,
- stisknuté klávesy,
- snímky obrazovky a
- další důležité informace.

Uživatel si následně může přehrát záznam ze sezení pomocí nástroje PyRDP, konkrétně programem pyrdp-player.

Kapitola 5

Implementace honeypotu

V této kapitole bude probrána implementace navrhnutých vylepšení honeypotu. Dále zde bude popsaná architektura celého systému včetně implementace skriptů na jednotlivých částech systému. U popisů jednotlivých skriptů nebude chybět ani popis odesílání, přijímání případně ukládání dat. V sekci 5.2 bude detailně charakterizována koncová stanice. V poslední sekci bude rozebrán skript, který všechna nasbíraná data ukládá do databáze, včetně schématu výsledné databáze.

5.1 Architektura systému

Jak již bylo zmíněno v sekci 4.4, architektura systému obsahuje celkem dvě cesty pro připojení útočníka na koncovou stanici. První cesta útočníka připojí přímo na koncovou stanici. Tato cesta má výhodu v tom, že útočník nepozná nastražený počítač, na rozdíl od cesty druhé, kde vstupuje do komunikace ještě PyRDP MITM jeho popis viz kapitola 4. Pokud je útočník zkušený, tak tento fakt může lehce odhalit viz sekce 2.4. První cesta sebou nese i zásadní nevýhodu, a to tu, že není možné získávat data o útočníkovi z PyRDP MITM.

Pokud se útočník rozhodne využít cesty bez PyRDP MITM, připojí se přímo na koncovou stanici, na níž mimo jiné běží program s názvem TCPDump. Po úspěšném přihlášení se spustí skript, který odešle data nasbíraná pomocí již zmíněného programu na server. Následně se přepne do druhého módu a začne zachytávat stisknuté klávesy, obsah schránky, otevřená okna a snímky obrazovky. Všechny tyto informace jsou ihned odesílány na server s PyRDP MITM pomocí protokolu TCP. Tento velmi jednoduchý protokol dokáže poslat všechna potřebná data zabalená ve zprávě, a to vždy ve formátu velikosti zprávy, typu zprávy a data. Na serveru s PyRDP MITM je připravený další skript.

Ten má za úkol přijmout data z koncové stanice a přidat k nim jméno útočníka. To se děje proto, aby bylo možné jednotlivé záznamy dohledat, uložit do databáze a následně data setřídit a dodat k nim potřebná metadata. Posledním krokem je odeslat data pomocí protokolu MQTT na server s databází. Na tomto serveru se nachází poslední skript, který má hned několik úkolů. Jeho hlavním úkolem je ukládat data do databáze tak, aby bylo jasné, jaká data patří k jakému útočníkovi. Druhým úkolem skriptu je obnovení koncové stanice.

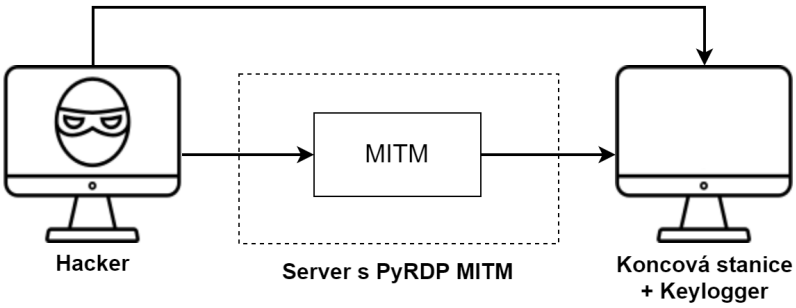
Pokud se útočník rozhodne využít druhou cestu, tak se k těmto datům připojí ještě data z PyRDP MITM s vylepšenou schránkou. Data mohou obsahovat dodatečné informace, a to včetně celého záznamu komunikace mezi útočníkem a koncovou stanicí. Tato data jsou odesílána na server s databází, kde se již zmíněný skript ukládá do databáze.

Jednotlivé skripty budou podrobněji popsány v podkapitolách, a to včetně detailního popisu koncové stanice, jednotlivých serverů a jejich komunikačních kanálů, schématu databáze a principu obnovení koncové stanice.

Stejně jako původní PyRDP MITM je i většina skriptů napsaná v programovacím jazyce Python 3. Přesněji jsou skripty napsané ve verzi Python 3.7.4[1]. Dalším využitým jazykem pro napsání skriptů je Visual Basic[2]. Skripty napsané v tomto jazyce slouží pro zapínání a vypínání ostatních skriptů/programů. Detailní popis v podsekci 5.2.3.

5.2 Koncová stanice

Koncová stanice, slouží jako bod, do něhož se musí útočník jakýmkoliv způsobem dostat, aby mohl získat cenná data nebo aby mohl využít tento počítač pro další útok. Koncová stanice je hlavním prvkem honeypotu s vysokou mírou interakce. Na koncovou stanici se lze připojit dvěma způsoby viz obrázek 5.1.



Obrázek 5.1: Možnosti připojení na koncovou stanici

5.2.1 Popis koncové stanice

Koncová stanice využitá v této práci obsahuje operační systém Windows 10 se všemi aktualizacemi. Všechny aktualizace jsou zde naistalovány proto, aby se útočník nemohl připojit pomocí zastaralých metod, ale jen pomocí nových, či útokem hrubou silou. Koncová stanice obsahuje také nainstalované různé programy, které jsou volně dostupné na internetu, například Gimp, LibreOffice, Adobe Reader a mnohé další. Ty jsou zde hlavně proto, aby dokreslily důvěryhodnost počítače. Následně jsou na pracovní ploše, a nejen tam, stažené různé obrázky a dokumenty, aby útočník získal pocit, že se napojil na úřední či domácí počítač. Všech těchto věcí si útočník může všimnout hned na první pohled. Koncová stanice ale obsahuje ještě další programy a skripty, o kterých útočník nemá žádné povědomí. Toto programové vybavení bude dále rozebráno. Co se týče hardwarového vybavení, je počítač vybaven 2 GB operační pamětí, procesorem Intel se dvěma logickými jádry o frekvenci 2.30 Ghz a integrovanou grafickou kartou. Vybavení tohoto počítače lze použít jako příklad minimální konfigurace stroje pro koncovou stanici. Dalším velmi podstatným bodem je rychlost Internetu. Pokud se počítač nachází v síti s pomalým či přerušovaným připojením, nemusela by odcházet všechna data včas, což by mělo za následek menší kvalitu sesbíraných dat.

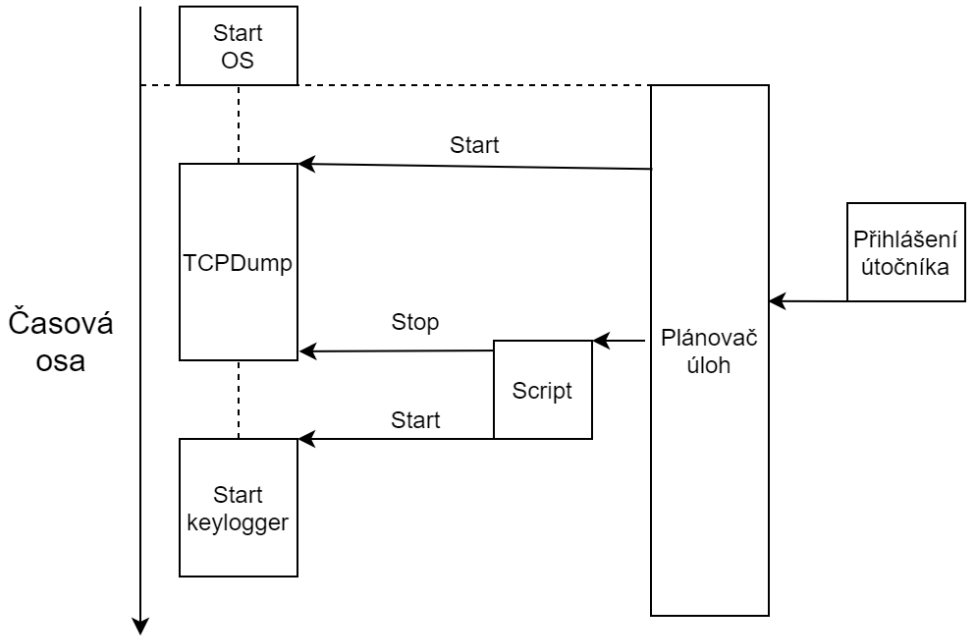
[1] https://www.python.org/
[2] https://en.wikipedia.org/wiki/Visual_Basic

5.2.2 Neviditelné programy pro sběr dat

Koncová stanice má za úkol odesílat důležitá data, proto musí obsahovat potřebné programy k tomu určené. Jak již bylo zmíněno, běží na koncové stanici program TCPDump, a to už od zapnutí počítače. Tento program běží v příkazovém řádku a jeho výstup se ukládá do předem připraveného souboru. Abych zajistil neviditelnost tohoto programu, byl program přejmenován a jeho vypnutí se nastavilo na akci přihlášení uživatele, takže program je pro útočníka neviditelný. Spuštění programu při startu počítače zajišťuje program Plánovač úloh, který je součástí operačního systému Windows. V něm bylo zapotřebí vytvořit úkol, který při zapnutí počítače spustí určitý program, v tomto případě je to příkazový řádek. Ten slouží jen ke spuštění programu TCPDump s určitými argumenty. TCPDump je na koncové stanici spuštěn kvůli získávání zaslaných paketů a IP adresy útočníka.

5.2.3 Neviditelné skripty

Nainstalované či spuštěné programy na koncové stanici pouze sbírají data. Jejich spuštění a vypnutí zajišťují skripty. Výše je uvedeno, že TCPDump se vypne při přihlášení uživatele. Toto vypnutí zajišťuje skript, jenž se spustí po přihlášení uživatele, a díky spojení programovacího jazyku Visual basic[3], plánovače úloh a služeb Windows je možné spustit skript na pozadí. Proto je možné nepozorovaně vypnout program TCPDump a zapnout skript pro čtení znaků z klávesnice a odesílání snímků obrazovky (dále už jen "Keylogger") viz obrázek 5.2.

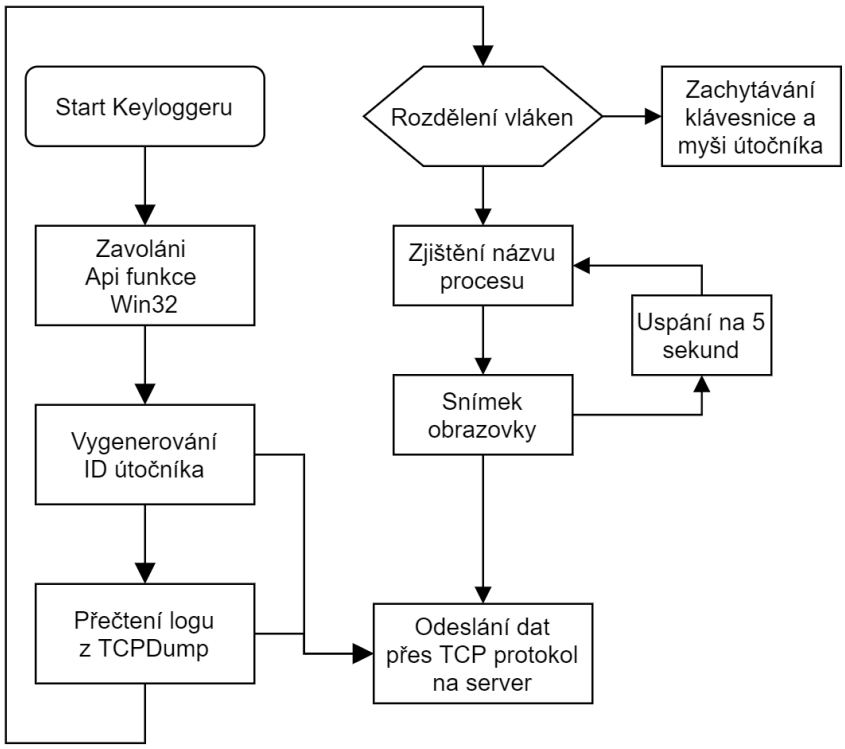


Obrázek 5.2: Diagram aktivity útočníka na RDP serveru

Keylogger po zapnutí zavolá API funkce Windows, které umožňují schovat proces ve správci úloh. Následně vygeneruje identifikátor útočníka a přečte log z programu TCPDump. Po získání IP adresy útočníka se odešle na server několik paketů s informací o iden-

[3] https://en.wikipedia.org/wiki/Visual_Basic

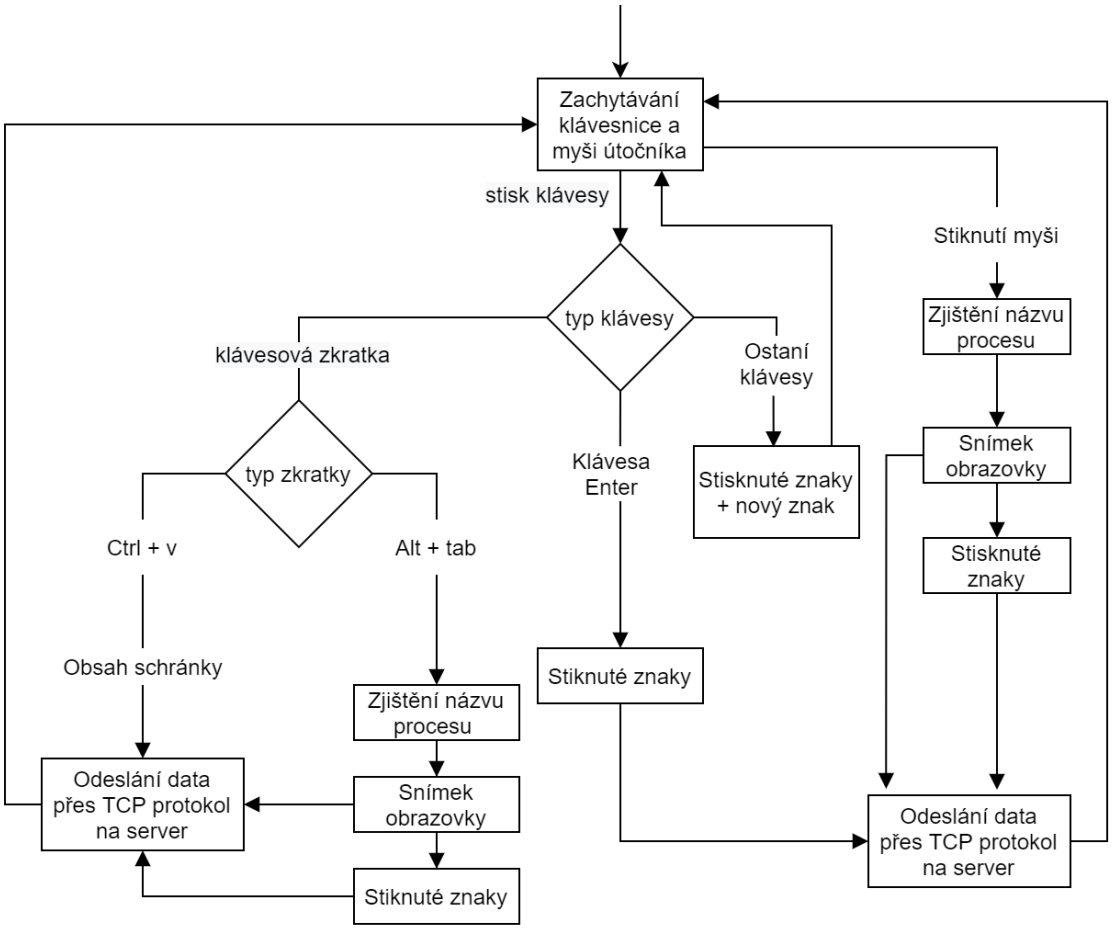
tifikátoru útočníka, času připojení, IP adresy a TCPdump viz diagram 5.3. Po odeslání těchto dat se skript přepne do sekundárního módu.



Obrázek 5.3: Vývojový diagram skriptu Keylogger část první

Keylogger se nejprve rozdělí na dvě vlákna. První vlákno má za úkol zachytit stisknuté klávesy a klávesové zkratky na klávesnici útočníka. Dalším úkolem je zachytit kliknutí myši na okna. Jestliže útočník stiskne klávesovou zkratku pro práci se schránkou, zavolá se obslužná metoda, která odešle obsah schránky na server pomocí TCP protokolu k dalšímu zpracování. Jestliže útočník stiskne klávesu Enter, okamžitě se zavolá obslužná metadata, která odešle na server stisknuté klávesy v aktivním okně, a to včetně názvu tohoto okna. Pokud útočník klikne myší na jiné okno, než je aktivní, automaticky se zavolá metoda pro zjištění názvu okna. Tato metoda zajistí také odeslání všech znaků, které byly zadány v posledním aktivním okně. Ihned za těmito stisknutými klávesami se zasílá informace o názvu okna a následuje snímek obrazovky pojmenovaný dle otevřeného okna. Druhé vlákno skriptu se stará jen o pořízení snímku obrazovky, a to každých pět sekund. Ty se následně odešlou pomocí TCP protokolu na server viz diagram 5.4. Server čeká přesně na tyto snímky, jakmile nepřijde snímek obrazovky či jiná data z koncové stanice déle než třicet vteřin, odešle se ze serveru ukončovací řetězec. Toto řešení slouží jako pojistka, kdyby útočník vypnul počítač, a tak z počítače nemohl odejít záznam o odpojení útočníka ze serveru. Na tento záznam čeká poslední skript (dále už jen "Saver"), který po tomto záznamu začne ukládat všechny nasbírané informace do databáze, případně vykoná potřebné operace pro konzistenci dat. Po uložení dat do databáze se zavolá API, který provede operace obnovení koncové stanice.

Všechna data se odesílají pomocí TCP protokolu přímo na server s PyRDP MITM. Protokol je navržen velmi jednoduše, obsahuje celkem tři části viz obrázek 5.5. Indexy



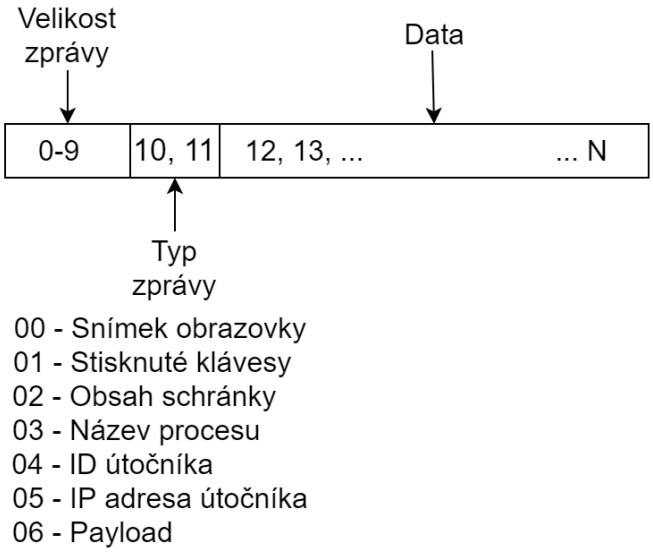
Obrázek 5.4: Vývojový diagram skriptu Keylogger část druhá

v paketu jsou po bytech kvůli možnosti zasílání velkých zpráv, které mohou obsahovat například soubory. První z částí je velikost zprávy. Ta zde musí být, aby bylo možné na druhé straně přečíst data správně. Druhá část je typ zprávy, díky této části či příznaku je na serveru možné rozluštit zaslanou zprávu. Poslední částí jsou samotná data. Protokol není nijak šifrovaný, ale na odesílání dat dostačující. Díky tomuto protokolu lze odesílat data z koncové stanice, i když ji má v rukou útočník. Pokud by útočník odhalil skript, jenž obsluhuje zasílání dat na server, nalezne zde pouze informace o IP adrese serveru, kam jsou data zasílána, což je jen další anonymní VPS v síti Internet.

5.2.4 Výhody odposlechu koncové stanice

Programy určené k odposlechu na koncové stanici přináší nový rozměr sběru dat do původního honeypotu PyRDP MITM. Původní honeypot se totiž zaměřoval pouze na sběr dat z MITM, s tímto rozšířením je možné získávat data od útočníků, kteří se připojili přímo na koncovou stanici. Z koncové stanice lze získat následující data.

Zaslaný kód Payload neboli zaslaný kód se odesílá na koncovou stanici před přihlášením uživatele. Payload tudíž může obsahovat škodlivý kód, který může využít nějaké zrani-



Obrázek 5.5: TCP zpráva

telnosti. Nejčastější zranitelností, kterou používají útočníci u protokolu RDP, je exploit BlueKeep. Dále může obsahovat přihlašovací údaje, jméno počítače, ze kterého se snaží útočník připojit, data aplikace a další důležitá data.

IP adresa Adresu lze získat pomocí událostí v operačním systému Windows nebo pomocí programu TCPDump, pro který jsem se rozhodl. Ze získané IP adresy je možné získat data, jako například ze které země je útok proveden, případně jestli se jedná o proxy nebo VPN. VPN a proxy servery umožňují vyšší stupeň soukromí a anonymní přístup k Internetu tím, že různými způsoby skrývají IP adresu [7]. Pokud útočník použije jeden nebo oba uvedené způsoby, je složitější odhalit pravou IP adresu útočníka. Existují i seznamy IP adres, jež jsou využívány jako VPN či proxy server. IP adresy z těchto seznamů lze zakázat, a tak omezit počet útočníků připojujících se do vnitřní sítě firmy.

Stisknuté klávesy Všechny klávesy včetně klávesových zkratek poskytují přehled o útočníkově taktice. Pokud útočník použije klávesovou zkratku pro operaci se schránkou, data ze schránky jsou odeslána na server. Tato data se následně dělí na soubory a text. Tímto způsobem lze získat URL adresy různých úložišť, kam si útočníci ukládají svoje skripty. Případně získat tento skript přímo z dat obsažených ve schránce.

Soubory a programy Netextová data lze získat dvojím způsobem. První způsob je z URL adresy, jež se ukrývá ve schránce, nebo v mysli útočníka. Jestliže útočník napíše, nebo vloží URL adresu do vyhledávače, tak je adresa následně odeslána na server ke zpracování. Druhý způsob spočívá v tom, pokud by se útočník snažil přenést skript pomocí schránky, tak po stisknutí klávesové zkratky, jež dovoluje práci se schránkou, se odešle obsah schránky na server a bude se zpracovávat stejným způsobem jako stažený soubor z URL adresy.

Přihlašovací údaje Získané údaje patřící k externím službám, jako je například vzdálené úložiště či email. Někteří útočníci totiž neváhají, otevřou si prohlížeč a přihlásí se do externí

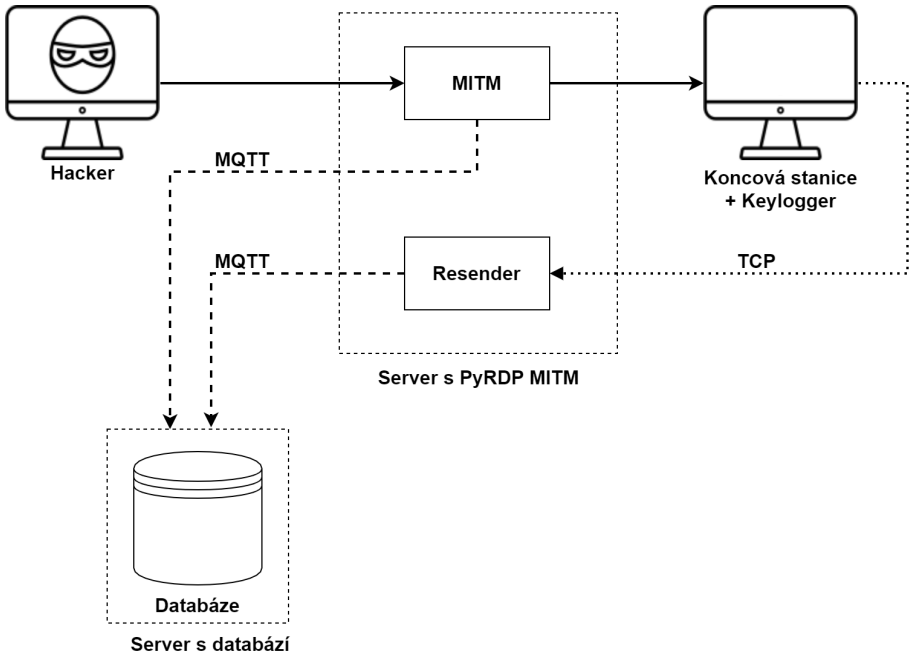
služby. Tím je od nich možné získat přihlašovací údaje do všech služeb případně získat jejich pravé jméno a další velmi zajímavé informace.

Snímky obrazovky Snímky se pořizují každých pět sekund a dokreslují útočníkovo počínání na koncové stanici. Některá nasbíraná data by bez snímku obrazovky mohla být špatně vykládána, což by mohlo mít za následek nepochopení útočníka. Tato situace je nežádoucí, a proto se všechny snímky ukládají do databáze. Snímky obrazovky se vytvářejí od připojení útočníka do jeho odpojení či vypnutí stroje, a to každých pět sekund. Snímky se vytvářejí také při přechodu na neaktivní okno. Tímto lze odchytit spuštěné skripty v příkazovém řádku.

Taktiku útoku Taktiku lze získat správným pochopením výše vypsaných dat. Na základě těchto dat lze rozpoznat útočníka od uživatele.

5.3 Server s PyRDP MITM

Server obsahuje instanci PyRDP MITM a skript pro přijmutí dat od koncové stanice (dále už jen "Resender") viz obrázek 5.6. Honeypot PyRDP MITM v rámci bakalářské práce byl vylepšen o komunikaci pomocí MQTT. Dále byl rozšířen o přenos souboru ve schránce. Pokud byl soubor přenesen ve schránce, je ihned odeslán na server s instancí Saver, kde je dále zpracováván. Skript s názvem Resender byl vytvořen pro přijetí dat pomocí TCP protokolu 5.5. Veškerá přijatá data jsou setříděna a následně odeslána přes MQTT.



Obrázek 5.6: Komunikace se serverem

5.3.1 Vylepšení PyRDP MITM

Původní honeypot přenášel skrz schránku pouze textová neformátovaná data, která se ukládala do logu. Po implementaci přesunu souboru ve schránce viz 4.3 byla funkce pro získávání obsahu schránky rozdělena na dvě části. První část získává textové řetězce z odeslaného paketu od útočníka. Pokud útočník vloží do schránky jen textová data, například URL či příkaz do příkazového řádku, jsou data vyhodnocena jako text, proto jsou vložena do logu a honeypot pokračuje dále ve své činnosti přeposílání paketu. Jestliže se útočník rozhodne přenést soubor, nejprve se v první části získá název souboru, ke kterému se následně připojí identifikátor. Po získání názvu souboru se funkce přepne do druhé části. Zde se získávají všechna potřebná data, jež definují obsah souboru. Data jsou následně ukládána do souboru. Při odpojení útočníka z koncové stanice se provede načtení záznamu útočníka, a to včetně všech souborů, které útočník přenesl skrz schránku. Záznam i soubory se odešlou přes MQTT na server se skriptem Saver. Data následuje záznam z logu, který obsahuje identifikátor útočníka, čas odpojení a celkový čas.

Jak již bylo zmíněno PyRDP MITM obsahuje log, do kterého se postupně zaznamenává průběh komunikace mezi útočníkem a koncovou stanicí. První záznam obsahuje jednoznačný identifikátor pro konkrétního útočníka, čas připojení, IP adresu a další položky. Vyjmenované položky jsou nejdůležitější pro další zpracování na posledním ze tří serverů v této architektuře. Další záznamy sebou nesou informaci o zaslaném kódu na server, přihlašovací údaje, textové řetězce ve schránce a další. Každý záznam obsahuje jednoznačný identifikátor útočníka, takže lze zpracovávat data i při paralelním připojení útočníků na server.

Při každém zápisu záznamu do logu se tento záznam odešle na server přes MQTT.

Restartování honeypotu PyRDP MITM Restartování není potřeba vykonat při každém připojení útočníka, restartovat PyRDP MITM je zapotřebí pouze za splnění určité podmínky. Podmínkou pro restart je přenos souboru či více souborů přes schránku, nastává zde totiž chyba nesprávného uvolňování části programu. Tato chyba nevzniká vždy, vzniká zhruba v 50 % případů. Od jednoho z vývojářů původního honeypotu jsem dostal informaci o tom, že při přenosu souboru končícího specifickým řetězcem nastává tato chyba v paměti. Poskytl jsem jim vylepšené zdrojové kódy s tím, že pokud by stihli opravit chybu, tak se mi ozvou. Pokud dojde k této chybě, poslední připojený útočník může s koncovou stanicí přes PyRDP MITM dále komunikovat. Po odpojení tohoto útočníka se další útočník už nepřipojí. Při snaze o připojení dostane hlášení o vnitřní chybě programu.

Restartování PyRDP MITM začíná až po odpojení útočníka. Při odpojení, jak již bylo zmíněno, se nejprve odešle záznam připojení a všechny přenesené soubory. Následně se počká deset vteřin kvůli odesílání těchto dat. Po uplynutí deseti vteřin se odešle ukončovací záznam, stejně jako všechna data, přes MQTT na server s databází. Následně se celý PyRDP MITM restartuje. Tato akce trvá přibližně pět sekund.

5.3.2 Dopady vylepšení PyRDP MITM

Vylepšení PyRDP MITM vedlo k lepšímu sběru dat, konkrétně souborů, které útočník přeposílá na server pomocí schránky. Dále vede k přehlednějšímu uložení dat v databázi, k níž je možné přistoupit pomocí programu určeného k prohlížení databází, například DBeaver[4]. Do databáze se ukládají pouze důležité informace o útočníkovi, jako je IP adresa, čas

[4] https://dbeaver.io/

a přihlašovací údaje. PyRDP MITM ukládá do logu i data, která nejsou pro další analýzu potřeba, jako je například senzor, message, level viz log 5.7.

První záznam

{"sensor": "PyRDP", "message": "New client connected from %(clientIp)s", "loggerName": "pyrdp.mitm.connections.tcp", "timestamp": "2019-11-10T09:47:14.391315", "level": "INFO", "sessionID": "Raymond469292", "clientIp": "147.229.206.75"}

Druhý záznam

{"sensor": "PyRDP", "message": "%(cookie)s", "loggerName": "pyrdp.mitm.connections.x224", "timestamp": "2019-11-10T09:47:14.391922", "level": "INFO", "sessionID": "Raymond469292", "cookie": "Cookie: mstshash=Microsoft"}

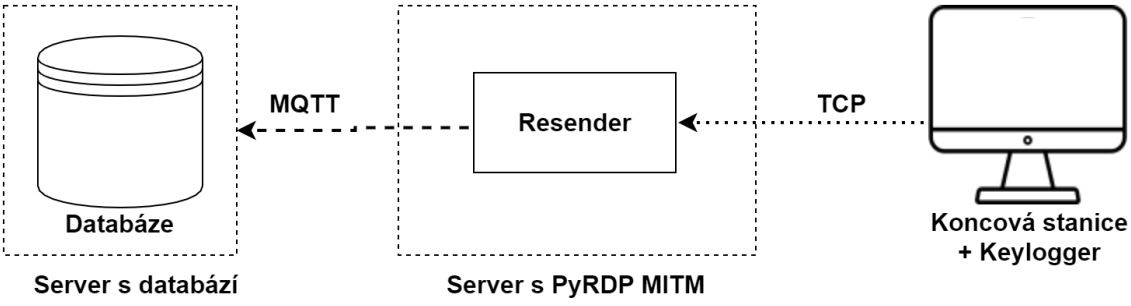
Obrázek 5.7: Log z PyRDP MITM

5.3.3 Popis skriptu Resender

Resender je skript určený pro přijetí TCP paketu na předem definovaném portu a následné přeposlání pomocí protokolu MQTT viz diagram 5.8. Spojením přijatých paketů vzniká zpráva, která obsahuje určitá data. Zpráva se následně rozdělí na tři části. První část označuje velikost zprávy, ta je potřebná pro úspěšné přijetí celé zprávy. Druhá část označuje typ dat, dle tohoto označení skript zpracuje data obsažená v poslední části zprávy viz obrázek 5.5.

Resender přijímá sedm typů dat. První dva typy jsou určeny pro nastavení skriptu a slouží také jako dodatečné informace potřebné pro správné odesílání dat. První typ dat obsahuje identifikátor útočníka, druhý typ nese informaci o názvu procesu.

Zbylé typy dat jsou určeny ke zpracování a následnému spojení s prvním typem dat. Ostatní typy dat jsou snímek obrazovky, schránka, stisknuté znaky, IP adresa a payload. Každý typ zprávy se zpracovává odlišným způsobem a následně je spojen s jednoznačným identifikátorem útočníka, aby bylo možné uložit tato data do databáze a v budoucnu mohlo dojít k dohledání všech informací k určitému útočníkovi.



Obrázek 5.8: Resender – přeposílání dat

První přijatá zpráva obsahuje pouze jednoznačný identifikátor útočníka, další zpráva obsahuje jeho IP adresu. Jakmile přijdou obě tyto zprávy, tak jsou spojeny a odeslány přes MQTT na server s databází, kde se dále zpracovávají. Následující zpráva obsahuje payload a stejně jako IP adresa se spojí s jednoznačným identifikátorem zaslaným v první zprávě a odešle se na server přes MQTT. Dále následuje kombinace zpráv, jako jsou stisknuté klá-

vesy, obsah schránky a snímky obrazovky. Obsah schránky funguje stejně jako IP adresa. U typu jehož obsahem je řetězec stisknutých kláves, se provádí ještě operace, která na začátek řetězce vloží název procesu. Název procesu se získá z aktivního okna na koncové stanici a odešle se pokaždé, když útočník přepne okno. Při přepnutí okna se odešle ještě snímek obrazovky. Pokud zpráva bude obsahovat snímek obrazovky, je zpracování takovéto zprávy náročnější na výkon. Snímek obrazovky se nejprve komprimuje a následně se o obrázku zjistí potřebné údaje jako velikost a název. Název snímku se skládá ze tří částí. První část je jednoznačný identifikátor útočníka, druhá část obsahuje název procesu stejně jako při zpracování řetězce znaků z klávesnice. Poslední část je náhodně vygenerovaný řetězec, ten slouží k jednoznačnému názvu snímku obrazovky.

5.3.4 Příspěvek skriptu Resender

Resender přispívá do celkové architektury tím, že přeposílá data z koncové stanice na zabezpečený interní server. Kdyby náhodu útočník odhalil Keylogger na koncové stanici, získá jen adresu serveru instancí PyRDP MITM a skriptu Resender. Tudíž nezíská adresu interního serveru. Dále konvertuje data na informace pomocí jejich spojování, případně získává metadata z přijatých dat. Jednotlivé informace jsou následně zasílány do databáze pomocí protokolu MQTT.

5.4 Server s databází

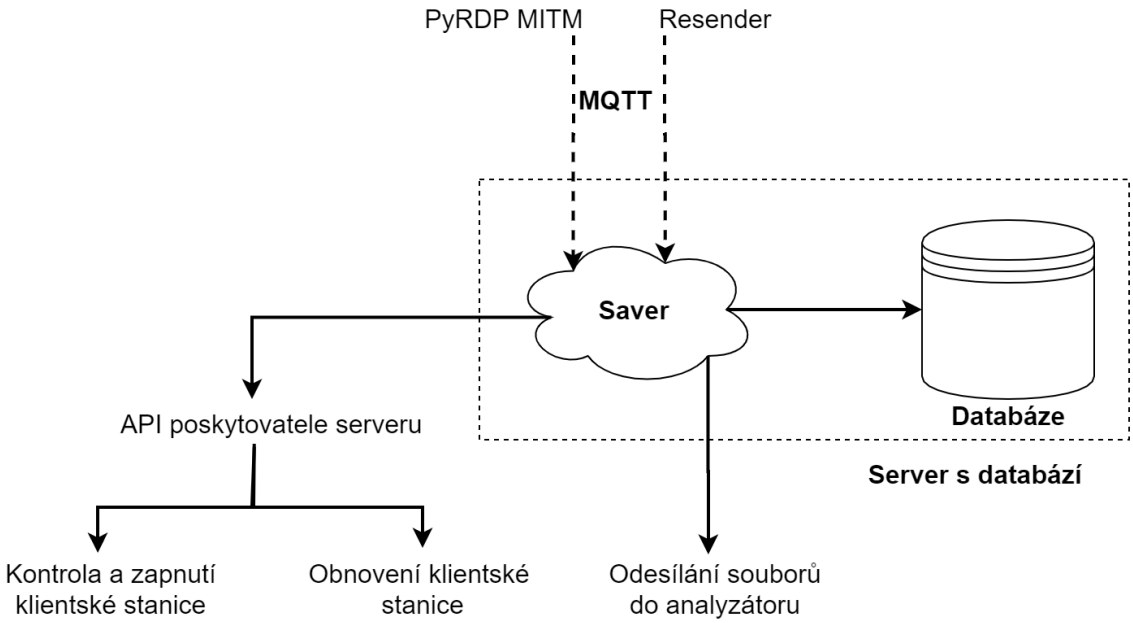
Poslední server je situován ve vnitřní firemní síti. Jeho hlavním úkolem je přijímat data z PyRDP MITM a skriptu Resender přes protokol MQTT a ukládat je do databáze ve vhodném formátu. Databáze se nachází na stejném serveru a její schéma bude podrobně rozebráno níže v podsekci 5.4.4. Jeho další úkoly jsou restartování koncové stanice, kontrola koncové stanice a odesílání získaných souborů do analyzátoru, viz obrázek 5.9.

5.4.1 Popis skriptu Saver

Jak již bylo zmíněno, Saver je skript nacházející se na posledním serveru a přijímá komunikaci pomocí protokolu MQTT. Po zapnutí se skript rozdělí na dvě vlákna. V prvním vlákně běží nekonečná smyčka, jež kontroluje stav koncové stanice. Pokud by se koncová stanice vypla zasláním škodlivého kódu, nebo kvůli chybě při startu obnovené koncové stanice, případně by byl vypnut hostující společnost, tak skript zajistí opětovné zapnutí koncové stanice.

Ve druhém vlákně jsou přijímána veškerá data a následně jsou vložena do třídy Attacker. Tato třída obsahuje všechny informace o útočníkovi. Před uložením do třídy data procházejí kontrolou a případnou úpravou, například konvertováním do správného formátu. Třída se skládá z několika údajů, které nelze obsáhnout v jedné zprávě, a proto je zapotřebí u všech přijatých zpráv kontrolovat identifikátor útočníka.

Jestliže útočník s tímto identifikátorem neexistuje, založí se nová instance této třídy a přijatá data se uloží do ní. Nová instance třídy se uloží do pole, aby byl možný pozdější přístup k této třídě. První zpráva s sebou nese mimo identifikační řetězec také čas připojení útočníka a jeho IP adresu. Při vzniku nové instance se zaznamená i aktuální čas, který slouží jako pojistka, kdyby nepřišla informace o odpojení útočníka. Pokud přijde zpráva obsahující známý identifikátor, jsou data ze zprávy vložena do vhodné existující instance podle identifikátoru. Tato operace se opakuje až do příchodu zprávy s ukončovacím řetězcem



Obrázek 5.9: Saver – obecný popis

nebo vypršením časového limitu. Poté jsou vložena poslední data ze zprávy do instance. Po úspěšném vložení poslední zprávy se začne celá instance postupně ukládat do databáze.

První se ukládá IP adresa s dodatečnými informacemi o její geolokaci. Následně se uloží do databáze základní informace o útočníkovi včetně cizího klíče do tabulky s jeho IP adresou, více viz 5.4.4. Při ukládání těchto dat se kontroluje, zda-li se útočník úspěšně připojil na koncovou stanici. Jestliže se úspěšně připojil, tak se vytvoří nové vlákno s obslužnou metodou, která kontaktuje API poskytovatele a požádá o obnovení koncové stanice. Mezitím se v původním vlákně dokončuje uložení dat do databáze. Po dokončení ukládání se instance odstraní z pole, které obsahuje jediného ukazatele na tuto instanci.

V případě, že útočník zaslal na koncovou stanici soubor či více souborů, budou všechny soubory uloženy do databáze a následně v novém vlákně odeslány k analýze do interního analyzátoru. Když útočník zkopíruje URL adresu na koncovou stanici, tak se URL adresa uloží do databáze, vytvoří se nové vlákno, ve kterém se provede kontrola URL adresy a případně zde dojde ke stažení souboru. Soubor se následně nahraje do vnitřního analyzátoru.

Saver obsahuje několik pojistek, kdyby došlo ke ztrátě informací o útočníkovi. První pojistkou je časová známka při vytvoření nové instance a při každém novém záznamu se kontroluje, zda-li nevypršel časový limit pro útočníka, který je nastaven na patnáct minut. Po uplynutí časového limitu je útočník automaticky uložen do databáze. Další pojistka je kvůli příchodu nekonzistenčních nebo opožděných zpráv. Pokud přijde zpráva bez předem určených klíčů, zpráva bude zahozena. Jestliže přijde opožděná zpráva bez užitečných informací, bude také zahozena.

Datový tok z PyRDP MITM

Saver zpracovává dva datové toky, první, jak již název napovídá, je datový tok z PyRDP MITM a druhý datový tok přichází ze skriptu s názvem Resender. V této sekci bude rozebrán datový tok z PyRDP MITM.

Zasílané zprávy mají z této instance většinou šest základních klíčů a jsou ve formátu JSON[5]. Z těchto klíčů jsou nejpodstatnější jednoznačný identifikátor útočníka a časová stopa. Záznamy dále obsahují dodatečné informace, jako jsou název senzoru, zpráva, název souboru odkud byl záznam zapsán a úroveň záznamu, viz obrázek 5.7. Poslední část záznamu poskytuje důležitá data, IP adresu, zkopírovaný text a další. Po příchodu první zprávy se vytvoří instance útočníka, která bude obsahovat jedinečný identifikátor, čas připojení, IP adresu a jestli se jedná o data z PyRDP MITM, nebo ze skriptu Resender.

Následující zpráva obsahuje informace o cookie, které útočník zaslal na koncovou stanici. Ty se posléze uloží do instance útočníka podle jeho identifikátoru. Další zprávy obsahující stejný identifikátor přinášejí data do stejné instance. Protože útočníci se mohou připojovat k serveru paralelně, je potřeba vždy kontrolovat jednoznačný identifikátor, aby byla data zapsána správnému útočníkovi. Poslední sada dat z PyRDP MITM obsahuje záznam útočníka, přenesené soubory a ukončovací zprávu. Ukončovací zpráva má stejné rysy jako všechny ostatní zprávy s rozdílem toho, že se zde nachází pole 'reason'. Jakmile zpráva obsahuje tento klíč, uloží se do instance útočníka čas odpojení a skript se přepne do druhé části, kde začne všechna data ukládat do databáze. Tento postup už byl vysvětlen výše.

5.4.2 Datový tok ze skriptu Resender

Druhý datový tok, jenž Saver musí paralelně zpracovávat, je ze skriptu s názvem Resender. Tok je odlišný v tom, že jeho data nejsou ve formátu JSON, ale zasílají se jako formát dictionary (slovník), tak jak je definován v programovacím jazyce Python 3. První zpráva sebou nese informaci o jedinečném identifikátoru útočníka, časové známce a IP adrese, stejně jako tomu je v předchozím případě. Další podobnost je, že zpráva sebou vždy nese jednoznačný identifikátor útočníka. Následující zpráva obsahuje payload, který byl získán pomocí programu TCPDump na koncové stanici a odeslán skriptem s názvem Keylogger. Hlavní odlišností od datového toku z PyRDP jsou příchozí snímky obrazovky. Ty nahrazují záznam, jenž odesílá PyRDP MITM. Snímky přicházejí na server každých pět sekund a mají specifický název. Název snímků, jak již bylo zmíněno, obsahuje identifikátor útočníka, název aktivního okna procesu, například Firefox, náhodný vygenerovaný řetězec a formát obrázku. Příkladem názvu je Clyde_firefox.exe_AUW3RVE.jpg. Novým typem zpráv jsou zprávy, jež obsahují klíč s názvem keyboard neboli klávesnice. Tento klíč ukazuje na data obsahující stisknuté klávesy útočníka, a to včetně názvu procesu, ve kterém byly stisknuty. Podobně jako v předchozím případě i zde je ukončovací zpráva. Ta obsahuje jen identifikátor a čas odpojení útočníka.

5.4.3 Příspěvek skriptu Saver

Největším příspěvkem je shromažďování důležitých dat ze dvou a více stanic do jedné databáze. Data tak mohou být rychleji analyzována. Dalším příspěvkem je obsluha API poskytovatele koncové stanice, která zajišťuje obnovu a dostupnost. To znamená že odpojení úspěšně připojeného a přihlášeného útočníka dojde k obnově koncové stanice, a proto se může připojit další útočník. Dalším příspěvkem je stažení souboru z URL adres, které útočník napsal, nebo zkopíroval na koncovou stanici.

[5] https://www.json.org/json-en.html

5.4.4 Popis databázového schématu

V této podkapitole bude podrobně rozebráno databázové schéma pro uložení potřebných dat. Databáze, jak již bylo zmíněno, se nachází na serveru ve firemní síti a využívá open source technologii PostgreSQL[6]. Do databáze ukládá data pouze skript s názvem Saver. Uživatel, jenž bude data analyzovat, může do databáze přistoupit pomocí terminálu nebo programu vhodného pro prohlížení relačních databází.

Nejprve budou v této sekci rozebrány datové typy použité při návrhu databáze a následně jednotlivé tabulky s příklady vložených hodnot. Následně zde budou popsány jednotlivé vztahy mezi tabulkami.

Databázové schéma

Databáze se skládá z deseti tabulek, z toho jedna slouží pouze k propojení útočníka s jeho soubory viz schéma 5.10. PostreSQL obsahuje datové typy pro specifické případy. Prvním z těchto typů, použitý při návrhu, je bigserial. Tento datový typ je určený hlavně pro primární klíč, protože v sobě obsahuje skript, který automaticky přičítá plus jedna k dalšímu záznamu. Také slouží jako jednoznačný identifikátor záznamu a je obsažen ve všech tabulkách tohoto schématu. Dalším datovým typem je inet. Tento typ je určený pro zápis IP adresy, a proto byl použit v první tabulce dle posloupnosti zapisování do databáze. Hlavním datovým typem pro zápis řetězců je text, případně jeho omezená varianta varchar. U varchar lze omezit textový řetězec na počet znaků. Datový typ určený pro zápis času či časové známky má jméno timestamp.

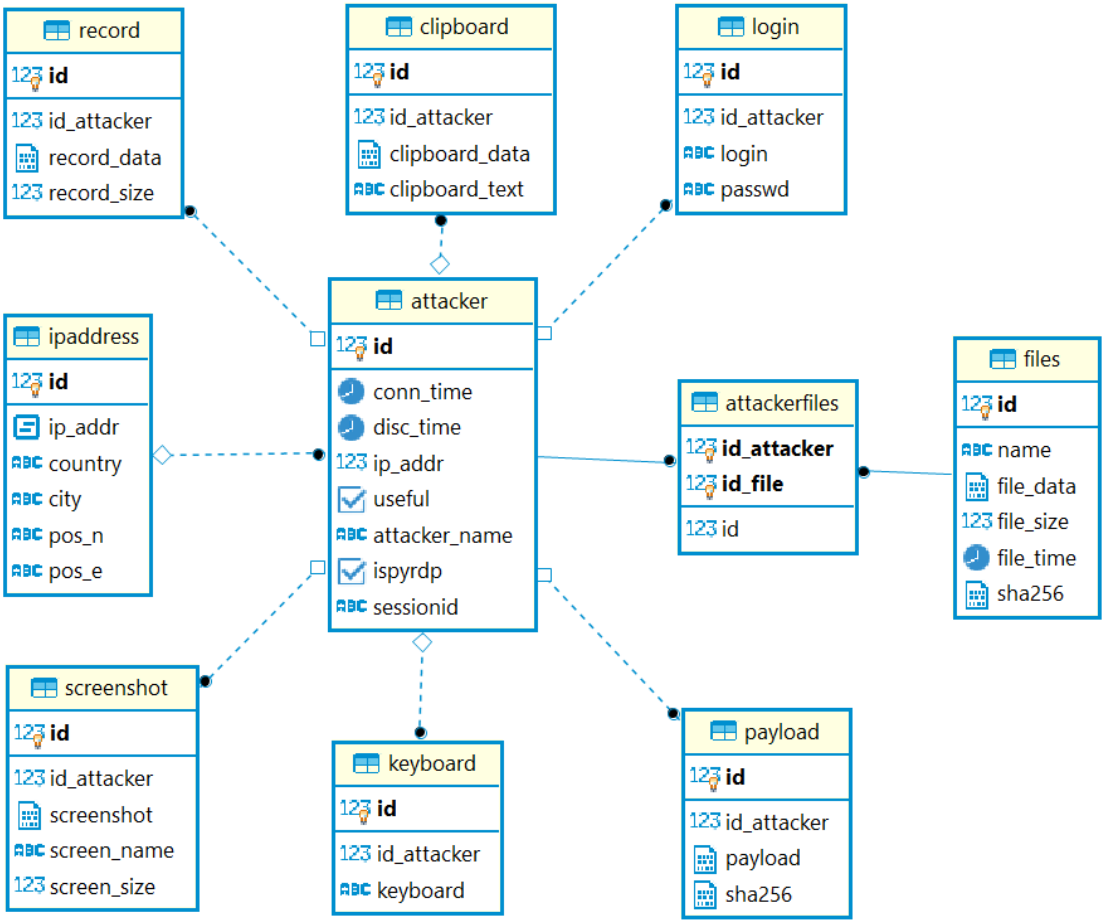
Poslední tři datové typy použité v této databázi jsou int, bool a bytea. Int je jednoduchý datový typ pro zaznamenané číselné hodnoty. Bool nebo boolean je typ pro uchování hodnoty pravda či nepravda. Posledním datovým typem je bytea. Tento datový typ slouží k uschování binárních dat, nejčastěji souborů či obrázku. Další databázové typy jsou k nalezení na oficiální stránkách[7].

Tabulka ipddress První tabulka dle posloupnosti zapisování je tabulka s názvem ipaddress. Tato tabulka obsahuje identifikátor (dále už jen id), IP adresu útočníka a geolokační údaje, jako jsou stát, město, zeměpisná šířka a výška, viz celé databázové schéma 5.10. Tabulka se skládá z šesti sloupců, které jsou zde rozebrány. Vždy je nejprve název sloupce, následně obsah daného sloupce a jeho datový typ.

- **id** – ID tabulky (bigserial),
- **ip_addr** – IP adresa útočníka (inet),
- **county** – stát ze kterého IP adresa pochází (varchar),
- **city** – město ze kterého IP adresa pochází (varchar),
- **pos_n** – zeměpisná šířka (varchar) a
- **pos_e** – zeměpisná délka (varchar).

Příkladem záznamu v této tabulce je následující 5.1:

[6] https://en.wikipedia.org/wiki/PostgreSQL
[7] https://www.postgresql.org/docs/9.5/datatype.html



Obrázek 5.10: Schéma databáze

id	ip_addr	country	city	pos_n	pos_e
35 980	186.122.57.83	Argentina	Hurlingham	-58.6336	-34.5962
47 527	191.209.21.237	Brazílie	São Paulo	-46.6322	-23.63
49 230	185.35.64.102	Francie	Paris	2.3281	48.8607

Tabulka 5.1: Příklad záznamu v tabulce ipaddress

Tabulka attacker Další tabulkou v posloupnosti ukládání je samotná tabulka attacker. V tabulce se nachází id, čas připojení a odpojení útočníka od serveru. V tabulce se také nachází cizí klíč do tabulky ipaddress, aby bylo možné k útočníkovi přiřadit správnou IP adresu. Dále tabulka obsahuje dva příznaky, první poskytuje informaci o užitečnosti a druhý o cestě, kterou si útočník vybral pro připojení na koncovou stanici. V tabulce se ještě nachází jméno, které útočník poslal při připojení na koncovou stanici pomocí cookie a jednoznačný identifikátor útočníka.

- **id** – ID tabulky (bigserial),
- **conn_time** – čas připojení útočníka (timestamp),
- **disc_time** – čas odpojení útočníka (timestamp),

- **ip_addr** – cizí klíč do tabulky ipaddress (integer),
- **useful** – příznak úspěšného připojení (boolean),
- **name** – přihlašovací jméno (varchar),
- **ispyrdp** – příznak, zdali se útočník připojil pomocí PyRDP MITM (boolean) a
- **sessionid** – jednoznačný identifikátor útočníka (varchar).

Příkladem záznamu v této tabulce je následující 5.2:

id	conn_time	disc_time	ip_addr	useful	name	ispyrdp	sessionid
30 238	2020-04-14 19:08:47	2020-04-14 19:18:48	30 354	False		True	Alvin855054
37 586	2020-04-17 13:19:25	2020-04-17 13:19:25	37 702	True		False	Alina
49 748	2020-04-20 09:10:43	2020-04-20 09:10:43	49 864	True	rdpscan	True	James141766

Tabulka 5.2: Příklad záznamu v tabulce attacker

Tabulka payload Následující tabulka v posloupnosti ukládání dat do databáze je tabulka payload. Ta obsahuje celkem čtyři sloupce, které obsahují informace o zaslaném payload na koncové zařízení, případně obsah cookie zaslaný z PyRDP MITM. V posledním sloupci je pak hash payloadu a to konkrétně hash sha256. Dále obsahuje cizí klíč do tabulky attacker 5.4.4.

- **id** – ID tabulky (bigserial),
- **id_attacker** – cizí klíč do tabulky attacker (integer),
- **payload** – kód/cookie zaslané při připojení na koncové zařízení (bytea) a
- **sha26** – hash hodnoty ve sloupci payload (bytea).

Příkladem záznamu v této tabulce je následující 5.3:

id	id_attacker	payload	sha256
5	36 502	payload	hash
13 243	49 740	Cookie: mstshash=Administr	hash
13 445	49 942	payload	hash

Tabulka 5.3: Příklad záznamu v tabulce payload

Tabulka login Čtvrtá tabulka v pořadí ukládání má název login. Obsahuje celkem čtyři sloupce, z nichž první je id a druhý je cizí klíč do tabulky attacker. Zbylé dva sloupce obsahují přihlašovací údaje, které používá útočník k úspěšnému přihlášení na koncovou stanici. Pro úspěšné přihlášení je zapotřebí zadat přihlašovací jméno a heslo. Na účet, který se nachází na vzdáleném počítači, se nelze přihlásit, jestliže účet není chráněn heslem.

- **id** – ID tabulky (bigserial),
- **id_attacker** – cizí klíč do tabulky attacker (integer),
- **login** – přihlašovací jméno (varchar) a
- **passwd** – přihlašovací heslo (varchar).

Příkladem záznamu v této tabulce je následující 5.4:

id	id_attacker	login	passwd
928	14 658	a<\b>Admin123	
987	46 542	Admin	admin12345.
988	46 542	Administrator	P@$$w0rd.

Tabulka 5.4: Příklad záznamu v tabulce login

Tabulka record Do této tabulky se ukládá pouze záznam útočníka, který se na koncovou stanici připojil přes PyRDP MITM. Stejně jako u předchozí tabulky jsou první dva sloupce id a cizí klíč útočníka. Předposlední ze čtyř sloupů obsahuje samotný záznam útočníka. Poslední sloupec obsahuje velikost záznamů. Záznam se neukládá vždy, ale jen tehdy, pokud má větší velikost než osmnáct kilobytů.

- **id** – ID tabulky (bigserial),
- **id_attacker** – cizí klíč do tabulky attacker (integer),
- **record_data** – záznam z PyRDP MITM (bytea) a
- **record_size** – velikost záznamu (integer).

Příkladem záznamu v této tabulce je následující 5.5:

id	id_attacker	record_data	record_size
9 789	32 088	Replay	7 612 197
9 784	23 096	Replay	10 478 905
9 778	22 472	Replay	14 399 939

Tabulka 5.5: Příklad záznamu v tabulce record

Table clipboard Tabulka pro zachytávání schránky se rozděluje na čtyři sloupce. Standardně jsou první dva sloupce id a cizí klíč do tabulky attacker. Další dva obsahují data přenesená ve schránce. Do těchto sloupců se zapisuje podle přijatých dat. Pokud útočník přenese skrz schránku soubor, jsou odeslána bitová data, která se ukládají do sloupce clipboard_data. Jestliže se jedná data textového typu, jsou uložena do sloupce clipboard_text.

Data se nikdy neukládají do obou sloupců najednou, ale vždy jen do jednoho. Nejcennější textová data jsou URL, příkazy a slovníky pro útok hrubou silou. URL adresa přenesená ve schránce se dále zpracovává za pomoci skriptu, který zajistí stažení souboru nebo webové stránky.

- **id** – ID tabulky (bigserial),
- **id_attacker** – cizí klíč do tabulky attacker (integer),
- **clipboard_data** – bytová data (soubory, programy) (bytea) a
- **clipboard_text** – textová data (URL, příkazy) (text).

Příkladem záznamu v této tabulce je následující 5.6:

id	id_attacker	clipboard_data	clipboard_text
9 615	19 115		http://s11.picofile.com/file/france.txt.html
9 725	37 586	Soubor	
9 725	46 542		Better!Better@Better#Better$ ...

Tabulka 5.6: Příklad záznamu v tabulce clipboard

V posledním řádku tabulky lze vidět část slovníku, jenž si útočník stáhl z webové stránky www.github.com a přenesl jej do programu NLBrute. Program NLBrute se používá k útokům hrubou silou na protokol vzdálené plochy. Slovník, který si útočník stáhl, využil v tomto případě jako seznam hesel a přidal k tomu přihlašovací jméno Administrátor. Útok byl kvůli obnově koncové stanice neúspěšný, protože trval jen krátkou dobu.

Table keyboard Tato tabulka slouží pro uložení všech stisknutých kláves útočníka. Skládá se celkem ze tří sloupců, z čehož první dva jsou id a cizí klíč do tabulky attacker 5.4.4. Do posledního sloupce se ukládají stisknuté klávesy útočníka. Záznam o stisknutých klávesách se vždy ukládá ve formátu název procesu a následně stisknuté klávesy v tomto procesu viz tabulka 5.7. Seznam stisknutých kláves se odešle z koncové stanice po tom, co útočník stiskne klávesu enter, nebo se překlikne do jiného okna.

- **id** – ID tabulky (bigserial),
- **id_attacker** – cizí klíč do tabulky attacker (integer) a
- **keyboard** – Stisknuté klávesy (text).

Příkladem záznamu v této tabulce je následující 5.4.4:

id	id_attacker	keyboard
69	35 308	#cmd.exe#PowerShell -command "$cmd = (New-//195.74.76.233 ...
115	37 586	#iexplore.exe#edific.com.mx/15.txt
131	47 010	#firefox.exe#Key.num_lockwhoewhoer.net

Tabulka 5.7: Příklad záznamu v tabulce keyboard

Ve sloupci keyboard lze na posledním řádku vidět zmáčknutí nealfanumerické klávesy, a to klávesy s názvem Numlock. Tato klávesa slouží k zapnutí číselné klávesnice. Tyto speciální klávesy jsou součástí uložených dat, aby bylo lépe poznat útočníkovo chování. Další klávesy které se mohou vyskytovat v záznamu, jsou například Ctrl, Alt, Esc.

Table screenshot Jedna z největších tabulek v databázi slouží k uchování snímků obrazovky. Tabulka obsahuje celkem pět sloupců. První dva sloupce jsou jako obvykle id a cizí klíč do tabulky útočník. V dalším sloupci jsou uloženy snímky obrazovky v komprimovaném formátu. Následný sloupec obsahuje jméno snímku. Jméno se skládá celkem ze čtyř částí. Na prvním místě se nachází identifikátor útočníka. Řetězec znaků pokračuje oddělovačem a jménem procesu. Řetězec je ukončen oddělovačem, náhodným řetězcem a formátem snímku. Poslední sloupec obsahuje informace o velikost snímků.

- **id** – ID tabulky (bigserial),
- **id_attacker** – cizí klíč do tabulky attacker (integer),
- **screenshot** – snímek obrazovky (bytea),
- **screen_name** – název snímku (varchar) a
- **screen_size** – velikost snímku (integer).

Příkladem záznamu v této tabulce je následující 5.8:

id	id_attacker	screenshot	screen_name	screen_size
23 5917	18 857	snímek	Nakia_mmc.exe_723P7VB.png	232 010
23 6039	19 115	snímek	Jason_n.exe_K58I3RS.png	1 318 243
23 7256	49 942	snímek	Shirley_explorer.exe_SBPYISO.jpg	122 136

Tabulka 5.8: Příklad záznamu v tabulce screenshot

Table files Poslední tabulka obsahující data má název files. Slouží pro uchování souborů, které útočník přenesl na koncovou stanici. Tabulka obsahuje celkem šest sloupců, z čehož první dva jsou id a název souboru. Následující sloupce obsahují data souboru, jeho velikost, čas pořízení a jeho hash SHA-256.

- **id** – ID tabulky (bigserial),
- **id_attacker** – cizí klíč do tabulky attacker (integer),
- **file_data** – data souboru (bytea),
- **file_size** – velikost souboru (integer),
- **file_time** – čas pořízení souboru (bytea) a
- **sha256** – SHA-256 hash souboru ve sloupci file_data (bytea).

Příkladem záznamu v této tabulce je následující 5.9:

Table attackerfiles Poslední tabulka celého databázového schématu má název attackerfiles. Jak již název napovídá, jedná se o útočníkovy soubory. Tato tabulka slouží k propojení mezi útočníkem a jeho soubory. Obsahuje dva sloupce fungující jako cizí klíče do tabulky attacker a files, které byly přeneseny na koncovou stanici.

- **id** – ID tabulky (bigserial),

id	name	file_data	file_size	file_time	sha256
10	n.exe	data	2 356	2020-04-03 21:38:23	hash
60	4815548run.bat	data	48	2020-04-09 13:48:02	hash
67	8043019config.ini	data	301	2020-04-14 14:10:46	hash

Tabulka 5.9: Příklad záznamu v tabulce files

- **id_attacker** – cizí klíč do tabulky attacker (integer) a
- **id_files** - cizí klíč do tabulky files (integer).

Příkladem záznamu v této tabulce je následující 5.10:

id	id_attacker	id_file
1	1 307	10
60	15 746	60
66	29 746	67

Tabulka 5.10: Příklad záznamu v tabulce attackerfiles

Relační vztahy Centrálním prvkem databázového schématu je tabulka attacker viz obrázek 5.10. Ta je propojena s tabulkou ipaddress pomocí cizího klíče, který je vložený do již zmíněné tabulky. Zbylé tabulky, mimo tabulky files obsahují cizí klíč do tabulky attacker. Při znalosti záznamu v této tabulce lze vytvořit SQL dotaz pomocí jeho id, a tak zjistit všechny potřebné údaje o útočníkovi. Jak bylo zmíněno tabulka, tak files obsahuje cizí klíč, a tudíž se nelze dotázat pomocí SQL dotazu přímo. Je potřeba nejprve zjistit v tabulce attackerfiles id útočníka a id souboru.

Kapitola 6

Analýza dat

Veškerá data, jež budou v této kapitole analyzována, se nachází v databázi. Jedná se o data, která jsem s využitím implementovaného honeypotu zachytil v průběhu dvou týdnů od 17.04.2020 do 30.04.2020. Data se do databáze nahrávají ze dvou rozdílných zdrojů a proto se i analýza bude dělit podle těchto zdrojů. Nejprve bude v této kapitole rozebráno, jaká data, ze kterého zdroje pochází, a následně budou data analyzována podle určitých kritérií. V poslední části této kapitoly bude rozebrán jeden útočník z každého zdroje.

6.1 Zdroje a jejich data

Celkem existují dva zdroje odkud putují data do databáze. První se nachází na jiném serveru a funguje stejně jako známý útok Man-In-The-Middle. Druhý zdroj se nachází na koncové stanici a odesílá data přímo z pod rukou útočníka. Pokud se útočník připojí přes PyRDP MITM, tak se do databáze ukládají data z obou zdrojů.

6.2 Rozbor dat

V této podkapitole budou data z obou zdrojů rozebrána a rozdělena podle určitých parametrů. Nejdůležitější parametr je příznak useful. Pokud totiž není nastaven tento příznak, tak neproběhlo úspěšné přihlášení útočníka na server a sesbíraná data neobsahují skoro žádné užitečné informace. V podstatě bez příznaku useful se honeypot s vysokou mírou interakce degraduje na honeypot s nízkou mírou interakce. Rozdíl mezi nimi viz sekce 2.3.

6.2.1 Dle IP adresy

V této sekci budou útočníci rozděleni dle jejich IP adres do několika skupin. Nejprve budou rozděleni podle země původu. Dalším kritériem je přítomnost VPN nebo proxy serveru. Na toto rozdělení je použité API[1], které umožňuje detekci VPN a proxy serverů s přesností na 80 %[2]. Posledním rozdělujícím kritériem bude úspěšné připojení z těchto IP adres. V poslední části této sekce bude porovnání příchozích IP adres před vylepšením a po vylepšení.

Útoky na koncovou stanici a na protokol vzdálené plochy přicházejí nejčastěji ze serverů v USA viz tabulka 6.1. Vyobrazení všech států je uvedeno v grafu 6.3.

[1] https://ip.teoh.io/vpn-detection

[2] Na základě výsledků testů třetích stran je náš nástroj pro detekci VPN a proxy přesný asi na 80 %.

Počet připojení	Země
21 792	USA
12 026	Francie
7 092	Itálie
7 005	Anglie
4 267	Vietnam

Tabulka 6.1: Tabulka prvních pěti připojení podle četnosti

V další tabulce 6.2 jsou vyobrazeny nejpoužívanější IP adresy. IP adresy jsem vložil do dříve zmiňovaného API, abych zjistil, jestli se jedná o VPN, či proxy server. Po analýze IP adres jsem zjistil, že většina přicházejících útoků byla přes VPN a proxy servery sídlící v USA.

Počet připojení	IP adresa	Stát	VPN/Proxy
6 512	195.81.180.231	Itálie	Ano
5 568	51.91.13.204	Francie	Ne
5 116	217.38.38.226	Anglie	Ne
3 412	171.244.143.35	Vietnam	Ne
2 342	177.36.196.5	Brazílie	Ne

Tabulka 6.2: Tabulka nejčastějších připojení s informací o VPN a Proxy serverech

V poslední tabulce 6.3 v této sekci je porovnání nejpoužívanější IP adresy pro útok před a po nasazení vylepšení. Před nasazením vylepšení v době od 4.11.2019 do 17.11.2019 se útočníci připojovali přes VPN a proxy servery v Rusku. V době od 17.04.2020 do 30.04.2020 převažují IP adresy z USA. Další nejpoužívanější IP adresy viz tabulka.

IP před nasazením	Stát	IP po nasazení	Stát
109.237.111.247	Rusko	195.81.180.231	Itálie
185.175.93.106	Rusko	51.91.13.204	Francie
116.58.30.125	Pákistán	217.38.38.226	Anglie
221.132.29.234	Vietnam	171.244.143.35	Vietnam
193.188.22.146	Rusko	177.36.196.5	Brazílie

Tabulka 6.3: Tabulka nejčastějších připojení před a po vylepšení honeypotu

6.2.2 Dle připojení

Uložená data v databázi lze rozdělit na dvě části pomocí příznaků odkud tato data přichází. Zde je důležitý ještě příznak useful, protože zápis dat ze skriptu Keylogger probíhá pouze po úspěšném připojení. U PyRDP MITM se ukládají data i když se útočník pouze pokusí připojit, a proto by mohl být v těchto datech nepoměr. Následující tabulka bude obsahovat připojení dle jednotlivých zdrojů.

Jak lze vidět v tabulce 6.4, rozšíření celkové architektury vede k výrazně efektivnějšímu sběru dat. Většina útočníků se připojuje raději přímo na koncovou stanici než cestou

Datum od	Datum do	PyRDP MITM	Keylogger	Celkem
17.4.2020	30.4.2020	32	115	147

Tabulka 6.4: Počet úspěšně připojených za 14 dnů

přes PyRDP MITM. Data budou podrobně rozebrána v podkapitole 6.3, která pojednává o celkové statistice připojených útočníků.

6.2.3 Dle užitečnosti

Jak již název napovídá, v této sekci se rozdělí záznamy na užitečné a na záznamy se základními daty. Základní záznamy obsahují pouze IP adresu útočníka, časy připojení a odpojení, zaslaný cookie a jednoznačný identifikátor. Užitečné záznamy obsahují stejná data rozšířená o stisknuté klávesy, soubory, snímky obrazovky, záznam útočníka a další. Pro pochopení útočníkova chování jsou zapotřebí pouze data s příznakem useful. U těchto záznamů je totiž možné pochopit útočníkovy úmysly a zařadit ho do tříd dle jeho chování. Třídy útočníků jsou popsány níže. V tabulce 6.5 je vyobrazen poměr úspěšně přihlášených oproti neúspěšně přihlášených útočníků na koncovou stanici.

Datum od	Datum do	Počet úspěšně	Počet neúspěšně	Celkem
17.04.2020	30.04.2020	147	78 027	78 174

Tabulka 6.5: Počet připojení za 14 dnů

6.2.4 Dle dnů

Analýzou každodenních útoků na koncovou stanici jsem zjistil, že někteří útočníci používají stále stejné VPN, či proxy servery. Proto je potřeba si tyto adresy pamatovat a případně vytvořit blacklist[3], který bude obsahovat tyto adresy. Tímto způsobem lze do budoucna částečně chránit koncovou stanici. Servery obsluhující VPN, nebo proxy spojení vznikají a zanikají každý den, a proto samotný blacklist nestačí. Možné řešení jak omezit útoky je využít whitelist, ten funguje opačným způsobem než blacklist. Whitelist obsahuje IP adresy nebo doménová jména, ze kterých je možné se připojit na koncovou stanici. V následující tabulce jsou vypsány nejčastěji použité IP adresy pro útok na koncovou stanici. Vzorek je brán ze čtrnácti dnů, a proto je v tabulce 6.6 přesně čtrnáct dní. V posledním sloupci v tabulce je obsažena informace o tom, zda-li se jedná o VPN, či proxy server. Tato informace je získaná z API, které zde již bylo zmiňováno.

6.2.5 Rozdělení útočníků do tříd

Dle útoků zachycených na implementovaném honeypotu jsem útočníky rozdělil do jednotlivých tříd podle jejich vzorců chování.

- *Další útok*, používá koncovou stanici pro další útoky;

[3] Blacklist je soubor obsahují IP adresy nebo doménová jména, ze kterých se nelze připojit na koncovou stanici

Datum	Počet připojení	IP adresa	Stát	Proxy/VPN
17.04.2020	1 354	51.91.13.204	Francie	Ne
18.04.2020	4 214	51.91.13.204	Francie	Ne
19.04.2020	620	167.172.133.192	USA	Ne
20.04.2020	262	185.253.218.16	Ukrajina	Ne
21.04.2020	227	185.253.218.16	Ukrajina	Ne
22.04.2020	276	54.165.74.149	USA	Ano
23.04.2020	898	195.81.180.231	Itálie	Ano
24.04.2020	862	195.81.180.231	Itálie	Ano
25.04.2020	1 652	177.36.196.5	Brazílie	Ne
26.04.2020	837	195.81.180.231	Itálie	Ano
27.04.2020	920	195.81.180.231	Itálie	Ano
28.04.2020	1 490	195.81.180.231	Itálie	Ano
29.04.2020	1 074	171.244.143.35	Vietnam	Ne
30.04.2020	2 338	171.244.143.35	Vietnam	Ne

Tabulka 6.6: Tabulka nejčastějších připojení v jednotlivých dnech

- *Zadní vrátka*, na koncovou stanici umístí skript pro možné další připojení, případně programy pro sledování uživatele;
- *Hledač souborů*, Hledá užitečné soubory, případně si je stahuje a;
- *Pozorovatel*, Neprovádí žádnou činnost na koncové stanici.

Všechny tyto třídy budou dále rozebrány včetně příkladů v podkapitole 6.4, která pojednává o ukázkových případech těchto útočníků.

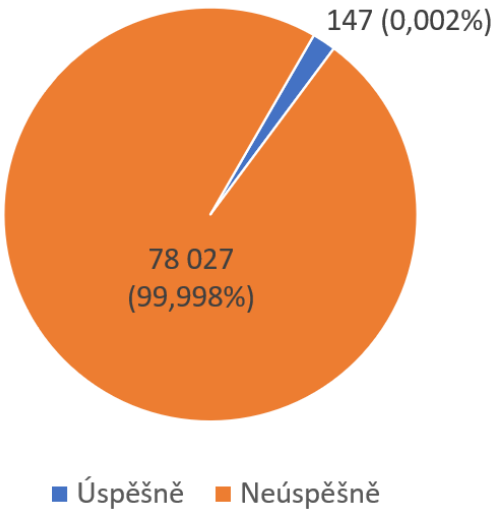
6.3 Statistické údaje

Následující sekce se bude zabývat statickými údaji z nasbíraných dat za sledovaný časový interval.

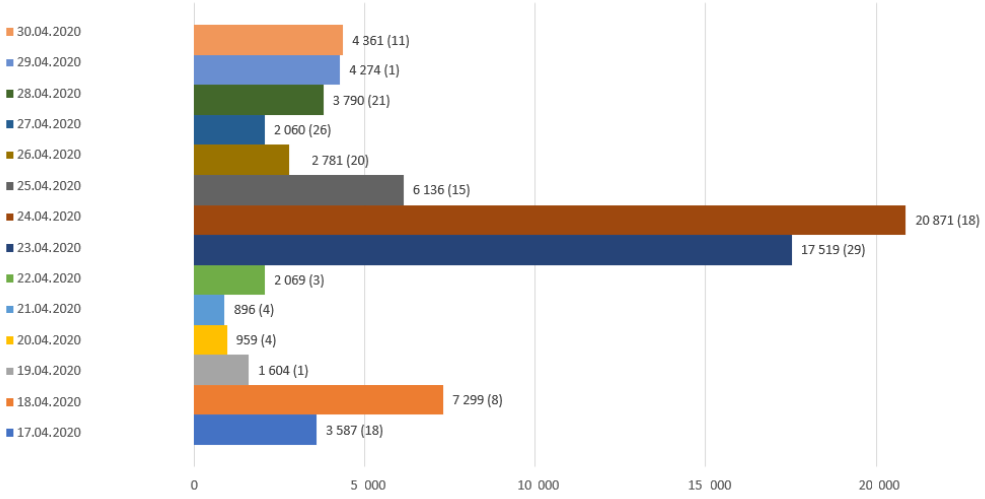
První graf viz 6.1 vyobrazuje poměr úspěšně k neúspěšně přihlášeným útočníkům na koncovou stanici za 14 dní pozorování. Počet úspěšně přihlášených je 147, cca 0,002 %. Toto procento je tak malé, protože jeden útočník může hrubou silou zkoušet velké množství připojení, než se připojí. Tyto pokusy se počítají do celkového pokusů o přihlášení.

Graf 6.2 zobrazuje počet připojených útočníků v jednotlivých dnech čtrnáctidenního pozorování. U každého dne se nacházejí dvě čísla, první označuje celkový počet připojených útočníků a druhé číslo označuje počet úspěšně přihlášených útočníků do systému. Nejčetněji se útočníci připojovali 23.04.2020 a 24.04.2020, nejméně se připojovali ve dnech 20.04.2020 a 21.04.2020. Jedná se hlavně o útoky hrubou silou, nikoliv o úspěšně přihlášené útočníky na koncovou stanici. Takových útočníků je minimum, cca 0,002 %, jak již bylo zmíněno.

Graf 6.3 obsahuje data o připojených útočnících z jednotlivých zemí. Nejvíce útoků přicházelo ze Spojených států amerických. Druhým státem v počtu útoků na koncovou stanici je Francie, která je následována Itálií a v těsném závěsu i Anglií.



Obrázek 6.1: Poměr úspěšně přihlášených útočníků k celkovému počtu pokusů



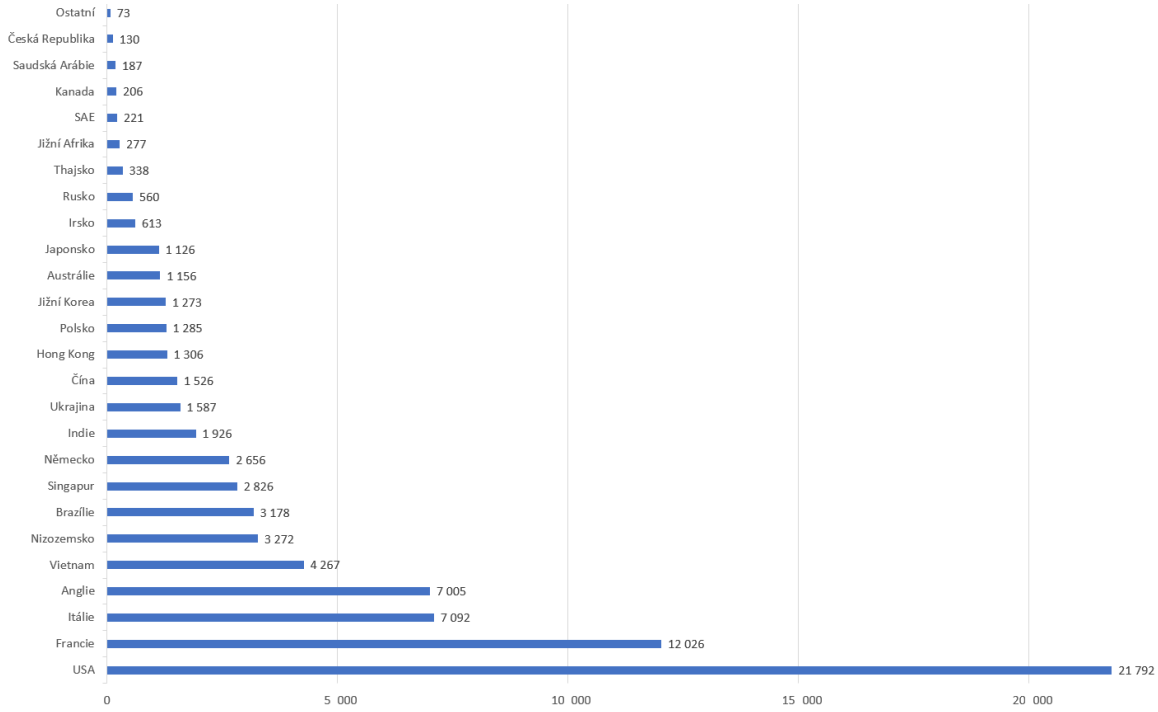
Obrázek 6.2: Graf – Připojení útočníci v jednotlivých dnech

6.4 Ukázkový případ útočníků

V této sekci budou rozebrány příklady úspěšně připojených útočníků. Někteří připojení útočníci mají podobné rysy chování a právě podle těchto rysů je lze rozdělit do jednotlivých tříd. Zástupci každé třídy zde budou podrobně rozebráni.

Další útok

Do třídy Další útok se řadí útočníci, kteří se připojí na koncovou stanici na přímo, nebo přes PyRDP MITM. Po připojení na koncovou stanici ihned vypínají antivirus. Někteří



Obrázek 6.3: Graf – Připojení útočníci z jednotlivých zemí

útočníci si vytvoří nový účet, nebo změní heslo k připojenému účtu. Následně otevřou webový prohlížeč a stáhnou si program NLBrute.exe[4].

NLBrute je program sloužící pro útok hrubou silou na další koncové stanice. Někteří útočníci si stahují program NLBrute i s IP adresami dalších obětí a slovník hesel. Po spuštění spojí program hesla s přihlašovacím jménem Administrátor a začne útočit na IP adresy se seznamu. Pokud soubor s programem NLBrute neobsahuje soubory s IP adresami a hesly, útočník si je stáhne z dalších stránek. Následně spustí útok a odpojí se z koncové stanice. Po odpojení útočníka nastane obnovení koncové stanice, a tudíž útok není nikdy úspěšný. Uložená data takového útočníka jsou v tabulce 6.7. Schéma tabulky je omezeno kvůli omezení stránky a přehlednosti.

id	conn_time	ip_addr	session id	screenshots
22 674	2020-04-11 18:28:05	54.165.74.149	Seema	data

id_attacker	keyboard	clipboard
22 674	http://s11.picofile.com/file/...	aqugrant.com/inc/n.exea

Tabulka 6.7: Ukázka uložených dat ze třídy Útočník

[4]https://tinyurl.com/y34ggtjz

Zadní vrátka

Útočník zařazený do této třídy ihned po připojení změní heslo a vytvoří si nový účet. Následně si stáhne svůj skript, který je nejčastěji umístěný na vzdáleném serveru. Stahování probíhá zpravidla vložením příkazu do příkazového řádku. Soubor se většinou uloží na neviditelné místo na koncové stanici. Příkaz vložený do příkazového řádku obsahuje skript, který má za úkol stáhnout soubor, rozbalit jej a následně spustit. Stažený soubor po svém spuštění vytvoří zadní vrátka do systému a dovolí útočníkovi útočit na další počítače. Případně je do systému nahraný Keylogger a podobné aplikace.

Honeypot obsahuje pojistku, která po odpojení útočníka zastaví všechny tyto útoky a obnoví koncovou stanici do původního stavu, aby se mohl připojit další útočník. Uložený záznam takového útočníka je následující:

id	conn_time	ip_addr	session id	screenshots
35 308	2020-04-16 04:50:26	178.62.96.159	Chester	data

id_attacker	clipboard	keyboard
35 308	#cmd.exe#PowerShell -command "$cmd...	178.62.96.159:8336/winapims

Tabulka 6.8: Ukázka uložených dat ze třídy Zadní vrátka

Hledač souborů

Jak již název napovídá, útočník zařazený do této třídy se připojí na server a začne prohledávat soubory. Nejprve prohledá celou plochu a následně prohledává známé složky, například obrázky a videa. Po ukončení prohlídky těchto souborů začne útočník spouštět nainstalované aplikace a zkouší hledat hesla. Jestliže najde nějaké zajímavé soubory, pokouší se je stáhnout, případně někam nahrát. Po dokončení všech těchto úkonů se útočník odpojí od koncové stanice. Tito útočníci zpravidla nejsou nijak zkušení a jejich příspěvek na pochopení taktiky útočníků je menší než u předchozích dvou tříd útočníků. Proto i uložená data obsahují jen základní informace o útočníkovi. Pár stisknutých kláves a snímky obrazovky, popřípadě záznam útočníkova počínání.

Pozorovatel

Útočníci v této třídě se pouze připojí na server, následně zavřou všechny aplikace a nic nedělají. Po nějaké časové době se odpojí. Na koncové stanici se nic nestane a útočník se ani o nic nesnaží. Do databáze se ukládají jen základní informace bez žádných stisknutých kláves.

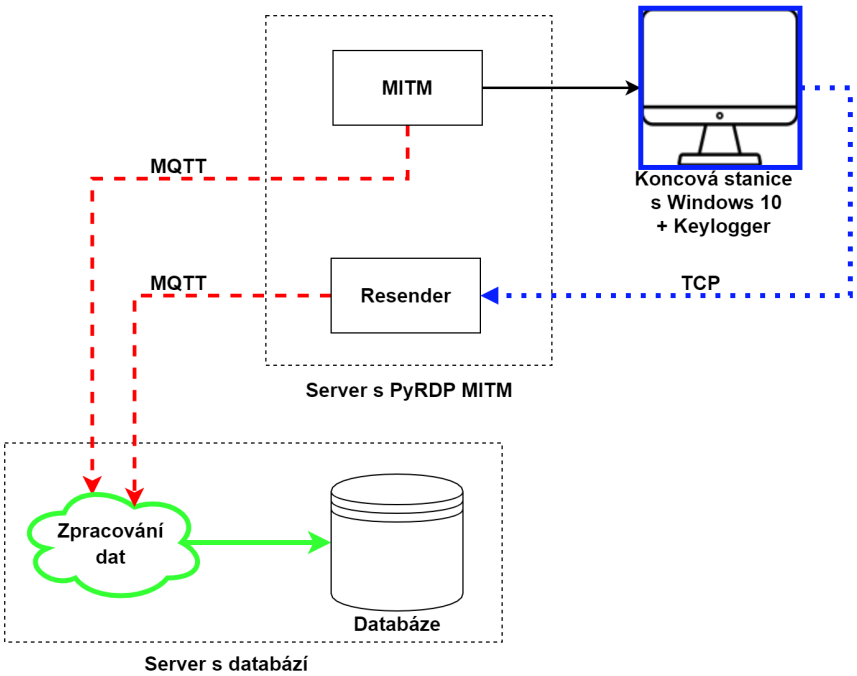
Třída Hledač souborů a Pozorovatel sbírají pouze základní data, která již byla vyobrazena v předchozích dvou tabulkách.

Kapitola 7

Testování honeypotu

Součástí každého projektu jsou testy, které testují program před spuštěním a nasazením na server. Jak již bylo zmíněno, tento honeypot obsahuje dvě cesty pro připojení útočníka na koncovou stanici a celý honeypot obsahuje čtyři skripty[1], které je nutné otestovat. Každý skript komunikuje s dalším skriptem nebo databází. Proto je důležité otestovat nejen funkčnost skriptu, ale i cesty mezi nimi. Každý skript má samostatnou složku a v této složce jsou umístěny testy na otestování daného skriptu.

Všechny testy jsou napsané v programovacím jazyce Python, ve verzi 3.7.4. Testy jsem napsal sám během implementace, a to proto, abych věděl, zdali je systém funkční a bez zbytečných chyb.



Obrázek 7.1: Honeypot s vyznačenými testy

Na obrázku 7.1 jsou barevně odlišeny testy jednotlivých skriptů. Modrá barva je testování skriptu s názvem Keylogger. Testuje se nejen samotný skript, ale i odesílání paketu

[1]Keylogger, Resender, PyRDP MITM, Saver

k dalšímu skriptu. Červená barva je pro testování Resender včetně protokolu MQTT. Součástí testu jsou i odeslaná data při testování Keylogger. Poslední zvýrazněnou barvou na obrázku je zelená. Ta značí testování skriptu s názvem Saver, který je poslední v této architektuře, i zde platí, že součástí testu jsou všechny předchozí testy, které budou dále detailně popsány. Celá architektura se pak testuje manuálně pomocí zkušebního připojení přes PyRDP MITM. Tento test prověří všechny cesty i skripty.

7.1 Popis testování

Testování probíhá ve dvou stupních. První test ověří funkčnost programu, pokusí se vytvořit všechny instance, naplnit je daty a odeslat je dalšímu skriptu. Pokud test projde úspěšně, vypíše na standardní výstup výsledek. Druhý test prověří chybně vložená data a zkoumá různé chyby. Přesný popis testu bude popsán níže. Skripty napsané v programovacím jazyce Visual Basic nejsou testované, protože celé skripty obsahují pouze pár řádků a testování probíhalo formou zkoušení funkčnosti manuálně.

7.1.1 Testování skriptu Keylogger

Keylogger je skript umístěný na koncové stanici a slouží k odesílání stisknutých kláves, obsahu schránek a snímků obrazovky. Z tohoto důvodu se testuje hlavně způsob odesílání a celkem jsou prováděny dva testy.

- Tento test je plně automatizován a testuje hlavně odesílací protokol TCP. Dále ověřuje správnou funkčnost načtení logu z programu TCPDump. Test nejprve odešle testovací identifikátor útočníka a z předpřipravených souborů načte IP adresu a log z programu TCPDump. Stejně jako při reálném nasazení na koncové stanici. Dalším krokem testu je odeslání názvu okna, za kterým následuje snímek obrazovky. Snímek obrazovky je důležitý s ohledem na další skripty v pořadí. Následně se zasílají data o stisknutých klávesách a obsahu schránky. Odeslaný obsah stisknutých kláves či schránky je v tomto skriptu irelevantní a slovo dostává až při ukládání do databáze.
- Další automatizovaný test prověřuje chybně vložené hodnoty, případně prázdné hodnoty. Nejprve odešle prázdné uživatelské jméno a pokusí se načíst log z programu TCPDump, který ve složce není. Následně se odešle snímek obrazovky beze jména. Po odeslání snímku se odešle prázdný obsah schránky a žádné stisknuté klávesy. Test se zakončuje získáním názvu aktuálního okna.

Odhalené chyby Při testování programu byla odhalena řada chyb. Většina z nich způsobovala odesílání nekonzistentních dat a jedna dokonce i pád programu. Další chyby jsou vypsány níže.

- Neošetřená chyba při načítání logu z TCPDump,
- pád programu při neaktivním okně,
- špatné pořadí odesílání a
- neodesílání souborů ve schránce.

Všechny tyto testy prověřují i skript s názvem Resender, protože tento skript se musí vypořádat s daty, jež byla odeslána pomocí protokolu TCP.

7.1.2 Testování skriptu Resender

Resender je skript sloužící pro přeposílání dat z koncové stanice na interní server pomocí protokolu MQTT. Z tohoto důvodu jsou testovány hlavně způsoby příchozí a odchozí komunikace. Celkem existují dva testy pro tento skript.

- Tento test prověřuje správné odesílání dat ze skriptu Resender přes protokol MQTT. Nejprve je vytvořena instance skriptu Resender, který se stará o odesílání dat a pak je postupně odeslán testovací útočník i se soubory. Tento test zároveň prověřuje i odesílání dat z PyRDP MITM.
- Úkolem tohoto testu je vyzkoušet přijetí dat, jejich zpracování a následné odeslání. V testu je nejprve navázáno spojení se skriptem Resender stejně jako u testu skriptu Keylogger a následně se odesílají data přes TCP protokol. Odesílají se jak očekávaná, tak neočekávaná data. U konce tohoto testu není odeslán ukončovací řetězec a čeká se na ukončení pomocí časového limitu.

Odhalené chyby Při testování skriptu Resender byly nalezeny chyby ve formě špatného formátu dat. První velká chyba nastala pro přijetí snímku obrazovky, který nebyl kompletní, respektive byl poškozený. Přesto se ho skript pokusil uložit a došlo tak k pádu skriptu. Druhý problém nastal při nehodě v datovém typu. V tomto případě to nevedlo k pádu skriptu, ale jen k neuložení názvu programu.

Stejně jako u testování skriptu Keylogger se testoval zároveň i skript Resender, tak při testování skriptu Resender se testuje i skript Saver.

7.1.3 Testování skriptu Saver

Saver je skript určený k uložení dat do databáze, tak aby nedošlo k nekonzistenci dat. Veškerá data přijatá přes protokol MQTT se nejprve uloží do třídy Attacker a následně je celá tato třída uložena do databáze viz kapitola 5.8. Z tohoto důvodu se bude testovat hlavně příchod zpráv a jejich následné uložení do databáze.

Test Testování tohoto skriptu se zaměřuje na ukládání dat do databáze včetně stažení souboru/skriptu z URL. Další činnosti, na které je test zaměřený, jsou obnovení koncové stanice a případné kontroly. Test nejprve vytvoří instanci útočníka se všemi potřebnými atributy a následně mu přiřadí všechna možná data. Poté provede činnost uložení této instance do databáze. Při ukládání aktivuje rekurzivní stažení souboru z URL a následné odeslání do vnitřního analyzátoru. Všechna data jsou postupně uložena do databáze včetně staženého souboru z URL.

Další testy nejsou potřeba, protože jsou již napsané pro testování skriptů Keylogger a Resender. Při každém testu těchto dvou skriptů byla odeslána data, která musel Saver zpracovat.

Odhalené chyby Po napsání všech testů se projevily chyby i ve skriptu s názvem Saver. Druhý test u skriptu Resender ukázal chybu při zjišťování informace ohledně IP adresy a její následné ukládání do databáze. Nebyla totiž zaslána IP adresa ale doménové jméno serveru. Další chyby viz níže.

- Stahování souboru z URL složené z IP adresy,

- špatné vložení do tabulky ipaddress vedlo k chybě skriptu a
- útočníci bez ukončovací zprávy nebyli uloženi do databáze.

7.2 Monitorování

Samotné testování pro udržení celé architektury v provozu nestačí, proto jsem rozšířil honeypot o monitorování. Jak již bylo zmíněno, honeypot obsahuje celkem čtyři skripty, koncovou stanici a databázi viz obrázek 4.3. Monitorování probíhá celkem na třech místech a pomocí webhooku aplikace Slack[2] se následné zprávy z monitorování ukazují v této aplikaci. Všechna chybová hlášení se zobrazují v aplikace Slack s červeným zvýrazněním po levém boku a obsahují informace o názvu skriptu, jenž byl monitorován, chybě která nastala a detailním popisu chyby.

Skript Resender Tento skript je monitorován pomocí skriptu napsaném v programovacím jazyce Python 3, který se spouští každých deset minut. Automatické spuštění zajišťuje Cron[3] na linuxové distribuci Debian. Utilita Cron funguje na velmi jednoduchém principu. Každý řádek obsahuje časový údaj o tom, kdy se má daný skript spouštět. Následuje cesta ke skriptu, případně absolutní cesta k interpretu skriptu[4]. Cesta k interpretu je pouze doporučená, narozdíl od cesty ke skriptu.

Monitorování tohoto skriptu se provádí pomocí TCP 3-Way Handshake. Monitorovací skript se snaží navázat komunikaci se skriptem Resender. Jestliže se nepovede komunikaci navázat, je zasláno hlášení pomocí webhooku do aplikace Slack. Chybové hlášení je doplněno o zvukové a vizuální upozornění.

Existují dva typy chybového hlášení. První chybové hlášení oznamuje, že daná IP adresa nebo port nejsou dostupné. Což znamená, že daná aplikace spadla. Druhý typ chybového hlášení oznamuje vnitřní chybu aplikace. Pokud dojde k takové chybě, aplikace stále běží, ale neodpovídá na navázání komunikace, tak se pomocí skriptu nedají přeposílat data z protokolu TCP na protokol MQTT.

Skript na monitorování obsahuje jen detekci chyby, pokud dojde k takové chybě, je potřeba skript ručně restartovat.

Kontrola koncové stanice Koncová stanice je kontrolována ze skriptu Resender, který si pamatuje poslední čas přijetí paketu od koncové stanice, přesněji od skriptu s názvem Keylogger. Pokud z Keylogger nedojde paket déle než šest hodin, zašle chybové hlášení do aplikace Slack pomocí webhooku.

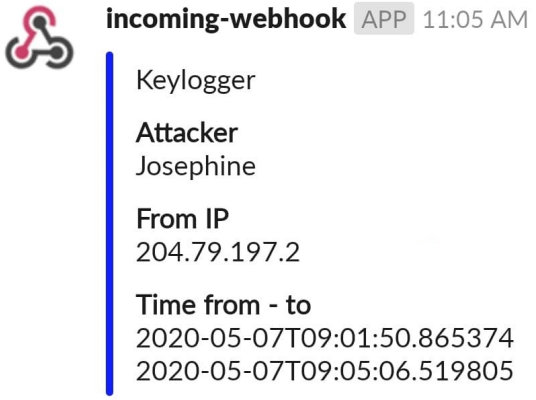
Jestliže nepřišel paket od koncové stanice déle než šest hodin, je velmi pravděpodobné, že koncová stanice není v provozu. Případně se nepovedlo její obnovení.

Kontrola PyRDP MITM Saver si stejně jako Resender pamatuje poslední přijatý paket. U Resender byl kontrolní čas šest hodin, zde je pouze deset minut. Tento čas vyplývá z analýzy dat, protože útočníci, kterým se nepovede úspěšně přihlásit na koncovou stanici, se připojují přibližně každou minutu, tudíž deset minut je dostačující čas na odhalení chyby.

Stejně jako u předchozích dvou skriptů se po zjištění chyby odešle chybové hlášení do aplikace Slack za pomoci webhooku.

[2] https://slack.com/intl/en-cz/
[3] https://en.wikipedia.org/wiki/Cron
[4] Interpret vykonává zvolený skript například python3 skript.py

Monitorování skriptu Saver Saver mimo kontroly skriptu PyRDP MITM odesílá i úspěšně přihlášené útočníky do systému. Zprávy mají na první pohled modré nebo zelené ohraničení na místo červeného. Zelené ohraničení mají útočníci, kteří se připojili přes PyRDP MITM. Naopak modré ohraničení mají útočníci, kteří se připojili a úspěšně se přihlásili na koncovou stanici. Ostatní útočníci se neodesílají. Zpráva mimo ohraničení sebou nese informaci o tom, ze kterého zdroje byl záznam získán, jednoznačný identifikátor, IP adresu útočníka, čas připojení a čas odpojení, viz obrázek 7.2.



Obrázek 7.2: Hlášení o připojeném útočníkovi

Zasílání chybových hlášení do aplikace Slack zlepšuje celkovou přehlednost o stavu celého systému. Po přidání zasílání zpráv o přihlášených útocích na koncovou stanici není potřeba neustálého nahlížení do databáze. Navíc doručování zpráv funguje i mimo vnitřní síť společnosti Avast, narozdíl od připojení do databáze, kdy musí být uživatel ve vnitřní síti, nebo připojený pomocí VPN.

Kapitola 8

Závěr

Tato práce je zaměřena na problematiku síťových útoků přes RDP a jejich řešení v podobě honeypotu. Je zde popsán způsob jejich využití, základní charakteristika a různé typy dělení. V následujících kapitolách je popsán protokol vzdálené plochy a honeypot PyRDP, který funguje nad stejným protokolem. Následující kapitoly se zabývají problematikou implementace, analýzy a testování včetně monitorování architektury.

Pro výslednou práci bylo potřeba nastudovat protokol vzdálené plochy [3], aktuální honeypot PyRDP [1] a další honeypoty.

Výsledná práce obsahuje návrh a implementaci architektury pro lepší sběr dat z PyRDP MITM a ze skriptu Keylogger. Tato data jsou následně ukládána do databáze v jednoduchém formátu. Důraz byl kladen na snadný a rychlý sběr dat z celkové architektury.

Nad nasbíranými daty je možnost provádět analýzu. Díky možnosti zaznamenávat útočníka, který se připojil přímo na koncovou stanici, jsou nasbíraná data věrohodnější a úplnější, než kdyby útočníci mohli přistupovat pouze přes PyRDP MITM.

Do nástroje PyRDP MITM jsem implementoval rozšíření o přenos souborů ve schránce a uložení jejich kopie na disk. Kvůli tomuto rozšíření může útočník přenášet jakékoliv menší soubory pomocí schránky na koncovou stanici. Přenášení souboru funguje obousměrně, takže si útočník může tímto způsobem stáhnout i data ze serveru. Dalším implementovaným rozšířením je podpora pro nejnovější protokol vzdálené plochy. Po implementaci navrhovaného rozšíření se zvedl počet úspěšně přihlášených útočníků připojujících se na koncovou stanici.

Další částí implementace je protokol MQTT a jeho začlenění do celkové architektury. Přínosem celé architektury je také centrální databáze na ukládání dat z PyRDP MITM a ze skriptu Keylogger. Díky skriptu Keylogger který je umístěn na koncové stanici, jsou nasbíraná data věrohodnější a úplnější, než kdyby útočníci přistupovali pouze přes PyRDP MITM.

Implementací vylepšení jsem dosáhl efektivnějšího sběru dat od útočníků. Nyní už není potřeba zásahu uživatele po připojení útočníka na koncovou stanici. Po navrhnutí jsem implementoval sběr souborů od útočníků nejen ze schránky, ale i z IP adresy či z URL, které útočník napíše, nebo jej přenese pomocí schránky. Díky tomuto vylepšení lze od útočníka sbírat skripty, které používá pro útok na koncové stanice.

Hlavním přínosem této práce je automatizace sběru dat z honeypotu s vysokou mírou interakce do vlastní databáze pomocí TCP a MQTT protokolu. Po nastavení koncové stanice a spuštění všech skriptů není potřeba už žádná interakce s honeypotem. Všechny následující časnosti jsou automatizovány.

Důležitým aspektem je i upozornění na úspěšně přihlášeného útočníka, případně na chybové hlášení o poruše architektury. Toto upozornění je odesláno do aplikace Slack, kde si jej může přečíst odpovědná osoba, a tudíž nemusí neustále kontrolovat jednotlivé skripty ani obsah databáze.

Do budoucna plánuji rozšíření Keylogger na koncové stanici o lepší sběr škodlivého kódu a automatické restartování nefunkčních skriptů. Následně plánuji architekturu rozšířit o vizualizaci dat pomocí webové stránky, kde budou veškerá data o útočníkovi včetně toho, do jaké skupiny je zařazen.

Literatura

[1] Bilodeau, O. *PyRDP is a Python 3 Remote Desktop Protocol (RDP) Man-in-the-Middle (MITM) and library.* [online]. 2019 [cit. 2019-12-30]. Dostupné z: https://github.com/GoSecure/pyrdp.

[2] Callegati, F., Cerroni, W. a Ramilli, M. Man-in-the-Middle Attack to the HTTPS Protocol. *IEEE Security & Privacy*. IEEE. 2. vyd. 2009, roč. 7, č. 1.

[3] Corporation, M. *[MS-RDPBCGR]: Remote Desktop Protocol: Basic Connectivity and Graphics Remoting* [online]. 2019 [cit. 2019-12-30]. Dostupné z: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rdpbcgr/5073f4ed-1e93-45e1-b039-6e30c385867c.

[4] Danchenko, N. M., Prokofiev, A. O. a Silnov, D. S. Detecting suspicious activity on remote desktop protocols using Honeypot system. In: IEEE. *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. 2017. ISBN 9781509048663.

[5] Fitzpatrick, D. *RDP on the Radar* [online]. 2019 [cit. 2019-12-30]. Dostupné z: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/rdp-security-explained/.

[6] Gonzalez Émilio a Labelle, F. *RDP Man-in-the-Middle – Smile! You're on Camera* [online]. 2018 [cit. 2019-12-30]. Dostupné z: https://www.gosecure.net/blog/2018/12/19/rdp-man-in-the-middle-smile-youre-on-camera/.

[7] Huang, S.-C. *Mobile VPN proxy method based on session initiation protocol.* Google Patents, říjen 12 2006. US Patent App. 11/099,508.

[8] IBM, D. A. S.-C. of a Eurotech), A. N. of Arcom (now. *MQTT* [online]. 2019 [cit. 2019-12-30]. Dostupné z: http://mqtt.org/.

[9] komárek, P. *Optimalizace nasazení honeypotů a logování událostí.* Brno, CZ, 2016. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Dostupné z: https://is.muni.cz/th/410240/fi_b/.

[10] Křoustek, J. *Protect yourself and your business from this troubling new trend* [online]. 2018 [cit. 2019-12-30]. Dostupné z: https://blog.avast.com/ransomware-attacks-via-rdp.

[11] Mokube, I. a Adams, M. Honeypots: concepts, approaches, and challenges. In: Adams, M., ed. *Proceedings of the 45th annual southeast regional conference*. 1. vyd. 2007, s. 321–326. ISSN 00002009.

[12] Němcová, J. *Analýza slovníkových útoků na honeypoty*. Brno, CZ, 2014. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Dostupné z: https://is.muni.cz/th/396396/fi_b/.

[13] Oosterhof, M. *Cowrie SSH / Telnet Honeypot* [online]. 2019 [cit. 2019-12-30]. Dostupné z: https://github.com/cowrie/cowrie.

[14] Schmieder, W. R., Srinivas, N. K., Hagiu, C., Abdo, N. Y., Stoyanov, V. K. et al. *User-mode based remote desktop protocol (RDP) encoding architecture*. Google Patents, květen 15 2012. US Patent 8,180,905.

[15] Spitzner, L. The honeynet project: Trapping the hackers. *IEEE Security & Privacy*. IEEE. 1. vyd. 2003, roč. 1, č. 2.

[16] Spitzner, L. Know your enemy: Honeynets. *Honeynet Project*. 2. vyd. 2005, č. 1.

[17] Týma, B. J. *Moderní honeypoty a honeynety*. Brno, CZ, 2018. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Dostupné z: https://is.muni.cz/th/buzbs/.

[18] Vestergaard, J. *Heralding - the credentials catching honeypot | The Honeynet Project* [online]. 2019 [cit. 2019-12-30]. Dostupné z: https://github.com/johnnykv/heralding.