Tel. 1843
El. p. cert@nksc.lt
https://www.nksc.lt

NACIONALINIS KIBERNETINIO SAUGUMO CENTRAS
PRIE KRAŠTO APSAUGOS MINISTERIJOS

INFORMACINIS BIULETENIS
ATVIRKŠTINĖ KENKĖJIŠKO KODO INŽINERIJA – III DALIS

2023 m. gruodžio 22 d.

Nacionalinis kibernetinio saugumo centras, vykdydamas kibernetinių incidentų tyrimus, esant poreikiui atlieka atvirkštinės kenkėjiško kodo inžinerijos veiksmus. Virusų, logerių, „Trojos arklių“ ir pan. kenkėjiško kodo kūrėjai stengiasi kaip įmanoma labiau paslėpti pėdsakus, užmaskuoti tam tikrus raktinius loginius elementus, sąsajų sąšaukas ir kodo identifikacijos elementus.

Pirmoje ir antroje dalyje aptarėme kelis pagrindinius kenkėjiško kodo (toliau - KK) analizės įrankius, bei kelis šiuo metu populiarius atakos vektorius, kuriais kenkėjas siekia savo tikslų tuo pačiu stengdamasis išlikti neaptiktas. Dažnais atvejais, kai kodas paprastesnis ar labiau automatizuotas/masiškas, pakanka anksčiau minėtų įrankių ir patikrinimo smėlio dėžėje (angl. *sandbox*). Be to, filtruoti pradinius atakos vektorius (pvz. URL, IP, *dropper/loader hash*) žymiai paprasčiau, nei aptikti sistemoje jau panaudotus eksploitus (angl. *exploit*) ir veikiančius kenkėjo įrankius, tokius kaip **reverse_TCP**, atsargines duris (angl. *backdoor*) arba *rootkit/bootkit*. KK įrankių aptikimo metodikos tyrimuose standartiškai kliaunasi registrų, žurnalinių įrašų ir procesų analize, tačiau šiame biuletenyje susipažinsime su tiesioginės dinaminės ir statinės, dar vadinamos hibridine, analizės metodikos pagrindais.

Ši metodika naudinga, kai turime KK paleidžiamąjį failą. Retesniais atvejais, kai turimas paleidėjas (angl. *loader*), o C2 serveris vis dar aktyvus ir su juo leidžiama susisiekti. Tuomet, galime atsekti žingsnius, kuriuos atlieka paleidėjas (angl. *loader*) ir parsiųsti nešulį (angl. *payload*), tačiau jo nepaleisti, o vietoje to pradėti atvirkštinę inžineriją.

KK aplikacijos, kurios buvo parašytos aukštesnio lygio (angl. *high-level*) kalba, tokia kaip C# ar Java gali būti lengvai dekompiliuojamos atgal į pseudokodą (angl. *pseudo-code*) ar netgi į kodą artimą originaliam išeities kodui (angl. *source code*). Tuo tarpu kai KK aplikacija parašyta žemesnio lygio kalba, tokia kaip C ar C++, lengvo būdo atvirkštiniam kompiliacijos procesui nėra ir tam reikalingas disasembleris (angl. *disassembler*).

Dėl didesnės nei įprasta apimties suskaidėme temą į tris dalis.

Biuletenių tipas – techninio pobūdžio. Šis informacinis biuletenis yra trečioji dalis.

<u>I dalis – analizės pagrindai</u>

- Analizės metodų apžvalga
- Statinės analizės pagrindai, pirminė analizė

Tel. 1843
El. p. cert@nksc.lt
https://www.nksc.lt

- Naudingi įrankiai

II dalis – pagrindiniai slėpimosi metodai

- Įkrovėjai (angl. *loaders*), pakeriai (angl. *packers*), droperiai (angl. *droppers*)
- Obfuskacija, šifravimas, kodavimas
- Polimorfinis, metamorfinis kodas

III dalis – gilioji analizė

- Dinaminės analizės įrankiai ir metodai
- Mišrios analizės atvirkštinė kenkėjiško kodo inžinerija

Pagrindiniai terminai

Dekompiliatorius (angl. *decompiler*) – programinės įrangos funkcionalumas leidžiantis transliuoti paleidžiamąjį failą į aukštesnio lygio atitikmenį, kuris būtų lengviau skaitomas žmogui, dažniausiai C-tipo sintakse.

Debugeris (angl. *debuger*) – programinė įranga skirta vykdyti instrukcijas pažingsniui su atminties ir proceso stebėjimu, sustabdymo pageidautinose vietose galimybe, bei daugeliu kitų naudingų analizės funkcijų.

Disasembleris (angl. *disassembler*) – programinė įranga išverčianti mašininį kodą atgal į aukštesnio – asemblerio lygio kodą.

Failų analizės įrankiai

Paskirtis	Įrankiai
Debuginimas, disasembleris ir dekompiliavimas KK statinei ir dinaminei analizei, komunikacijų perėmimui	IDA Pro, WinDbg, Ghidra, JustDecompile
Procesų monitoringas	Regmon, Process monitor, Process explorer
Atminties perėmimas, RAM nagrinėjimas	Task Manager, System configuration, ProcDump, Process Explorer, FTK Imager, Dumpit
Artefaktų tyrimas	Windows Event viewer, Registry Viewer, Autoruns (prieš ir po KK paleidimo)
PE failų deobfuskacija	de4dot, Ghidra
Skaitmeninių atvaizdų analizė, duomenų atstatymas	Autopsy, DMDE

Tel. 1843
El. p. cert@nksc.lt
https://www.nksc.lt

Aparatinės įrangos analizė	Hexdump, binwalk, firmadyne
Interaktyvios „smėliadėžės" (*online sandboxes*) ir skenavimo įrankiai	antiscan.me, urlscan.io, virustotal.com, Cuckoo

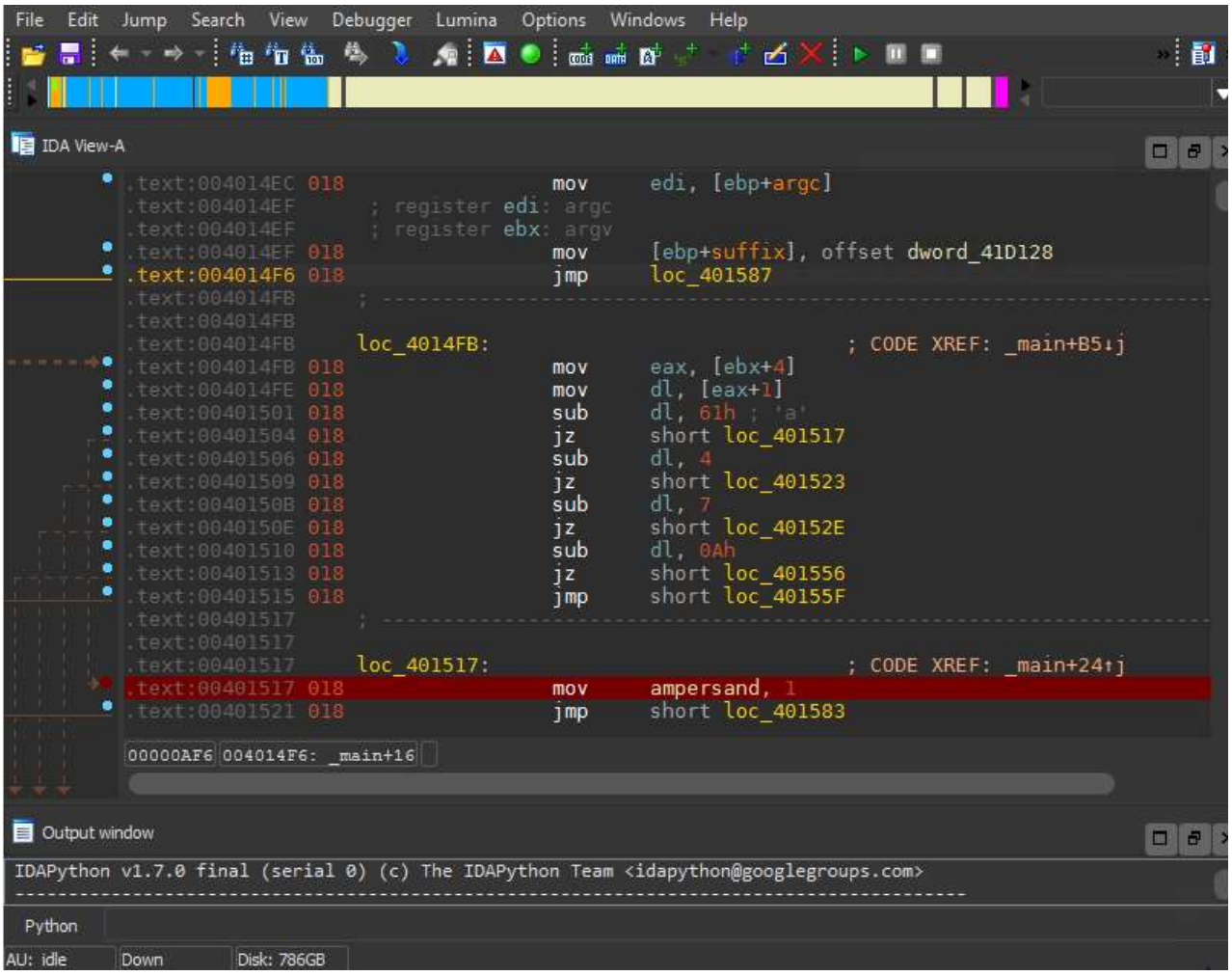
Hibridinės analizės įrankiai

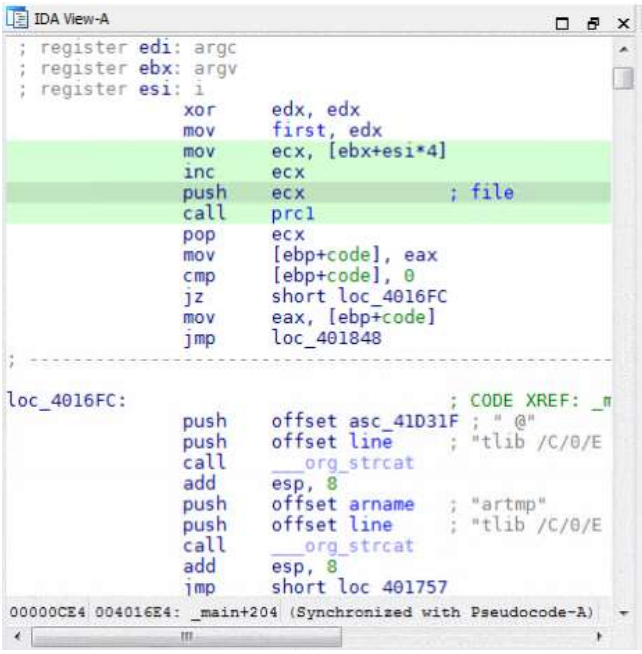
Disasemblerio programinė įranga analizuoja paleidžiamąjį failą ir konvertuoja sukompiliuotą kodą atgal į jo asemblerio reprezentaciją. Geriausi disasembleriai esantys rinkoje taip pat vizualiai suskirsto asemblerio kodą intuityviau jį atvaizduodami, t. y. blokais, funkcijomis, loginėmis šakomis ir pan. Įprastai disasembleris naudojamas kartu su debugeriu atvirkštinės inžinerijos ir sudėtingesnių eksploitų kūrimo metu. Toks hibridinis metodas apjungiant statinę ir dinaminę analizę padidina tyrimo efektyvumą. Industrijoje pagrinde naudojamas disasembleris yra IDA Pro, mažiau populiari jo alternatyva – Ghidra . Keletas mažų ir rečiau naudojamų disasemblerių – Binary Ninja ir Radare. Žemiau pateikiame šių įrankių įvertinimą.

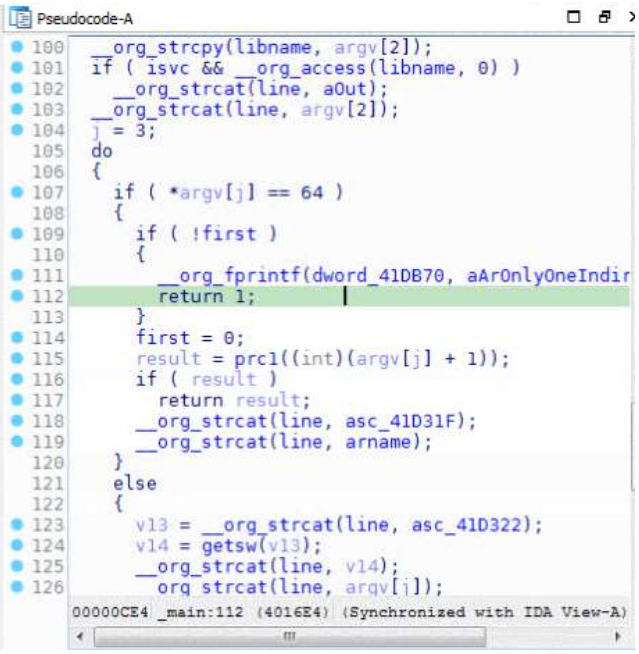
IDA Pro

Aukšta produkto kaina atspindi teikiamus privalumus - IDA Pro palaiko daugiau nei 60 procesorių šeimų, siūlo patogų UX ir UI dizainą, turi integruotą debugerį ir galingą dekompiliatorių šešioms skirtingoms platformoms. Dekompiliatorius atlikęs mašininio kodo analizę pateikia C stiliaus kodo reprezentaciją (angl. *pseudo-code*):

Tel. 1843
El. p. cert@nksc.lt
https://www.nksc.lt



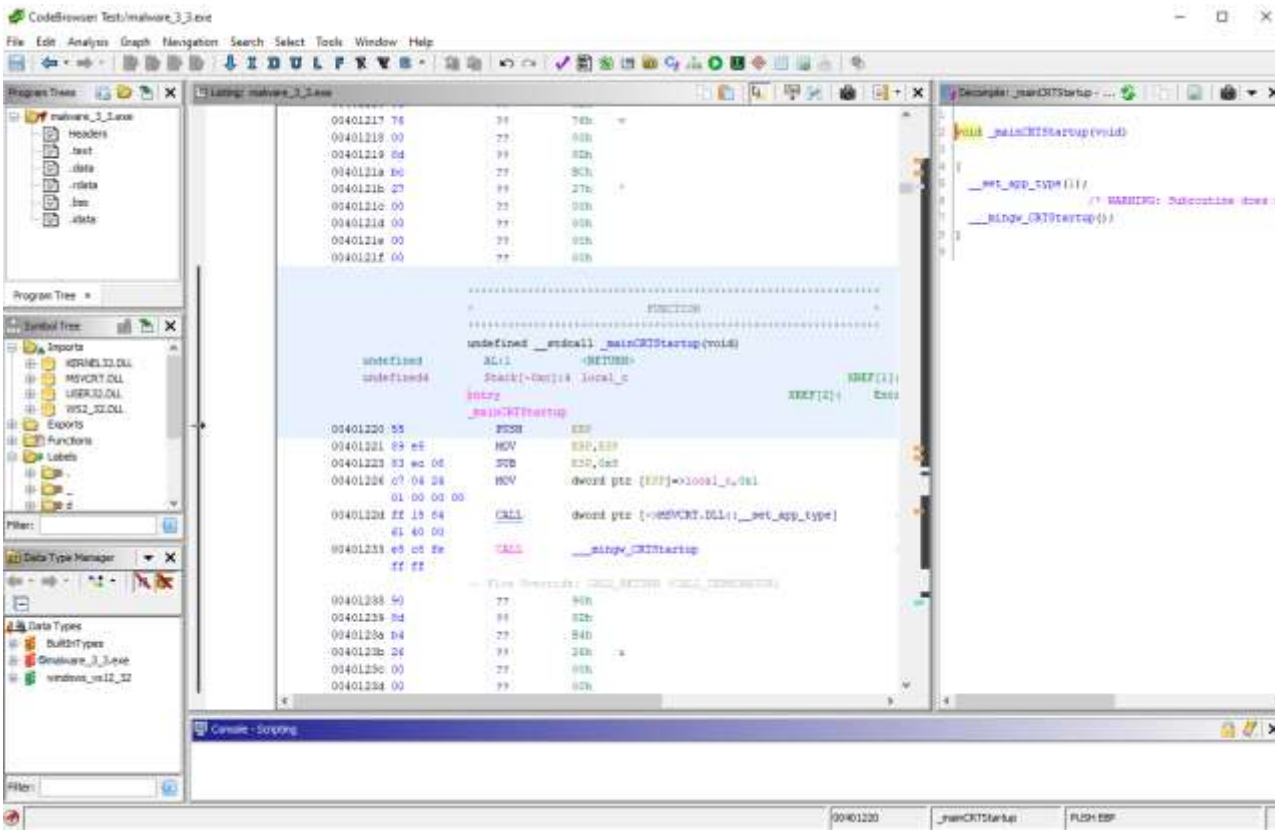




Tel. 1843
El. p. cert@nksc.lt
https://www.nksc.lt

Ghidra Software Reverse Engineering (SRE)

Ghidra yra nemokama, atviro kodo SRE sistema kurią sukūrė ir palaiko JAV Nacionalinio saugumo agentūros Tyrimų direktoriatas. Į sistemą įeina rinkinys aukštos kokybės analizės įrankių pritaikytų pagrindinėms platformoms – „Windows“, „macOS“ ir „Linux“. Nors, palyginus su IDA Pro, šis įrankis kai kuriais aspektais nusileidžia (pvz. UX ir dizainu), kitais atžvilgiais netgi pralenkia lūkesčius (įskiepių gausa). Ghidra palaiko dekompiliavimą, turi disasemblerį ir asemblerį, grafų ir skriptų aplinkas, su šimtais kitų savybių naudingų analitikos uždaviniuose. Ghidra gali būti paleidžiama interaktyviu ar automatizuotu metodu, taip pat turi platų spektrą palaikomų procesorių instrukcijų rinkinių.



Kiti dekompiliatoriai ir deobfuskatoriai

Dnspy (https://github.com/dnSpy/dnSpy)

ILSpy (https://github.com/icsharpcode/ILSpy)

Java Decompiler (http://java-decompiler.github.io/)

Tel. 1843
El. p. cert@nksc.lt
https://www.nksc.lt

DotDumper (https://github.com/advanced-threat-research/DotDumper)

JustDecompile (https://www.telerik.com/products/decompiler.aspx)

De4dot deobfuscator (https://www.kali.org/tools/de4dot/)

Phoenix Protector (https://ntcore.com/?page_id=384)

Saugioji praktika

Atliekant kenkėjiško kodo analizę svarbu tai daryti saugiai, izoliuotoje aplinkoje. Naudokite virtualizacijos įrankius su apribota tinklo prieiga, nepamirškite išsaugoti atvaizdų (angl. *snapshots*) po kiekvieno KK paleidimo darbines analizės aplinkas atstatyti į prieš tai buvusią būseną. Tokiu būdu išvengsite ilgalaikių padarinių, bei pavojingų incidentų, tokių kaip šoninio judėjimo (angl. *lateral movement*) ir privilegijų eskalavimo (angl. *privilege escalation*). Štai keletas analizei patogių aplinkų distribucijų atvaizdų ir kt. naudingų resursų:

Windows Malware Analysis Distribution (https://www.mandiant.com/resources/blog/flare-vm-the-windows-malware)

REMnux: Linux Toolkit for Malware Analysis (https://remnux.org/)

Kuruojamas kenkėjiško kodo įrankių sąrašas (https://0x1.gitlab.io/security/Malware-Analysis-Tools-List/)

Naudingų resursų sąrašas kuriuojamas SANS instituto (https://www.sans.org/blog/-must-have-free-resources-for-malware-analysis/)

Hibridinės analizės metodika

1. Saugioje aplinkoje parsisiunčiame KK mėginį (https://bazaar.abuse.ch/ ar https://github.com/fabrimagic72/malware-samples) ar naudojame jau turimą paleidžiamąjį failą, pavyzdžiui, aptiktą savo sistemoje.
2. Importuojame failą pasirinktame disasemblerio įrankyje (pvz. *Ghidra Code Browser*).
3. Paleidžiame dekompiliatoriaus analizę.

Tel. 1843
El. p. cert@nksc.lt
https://www.nksc.lt

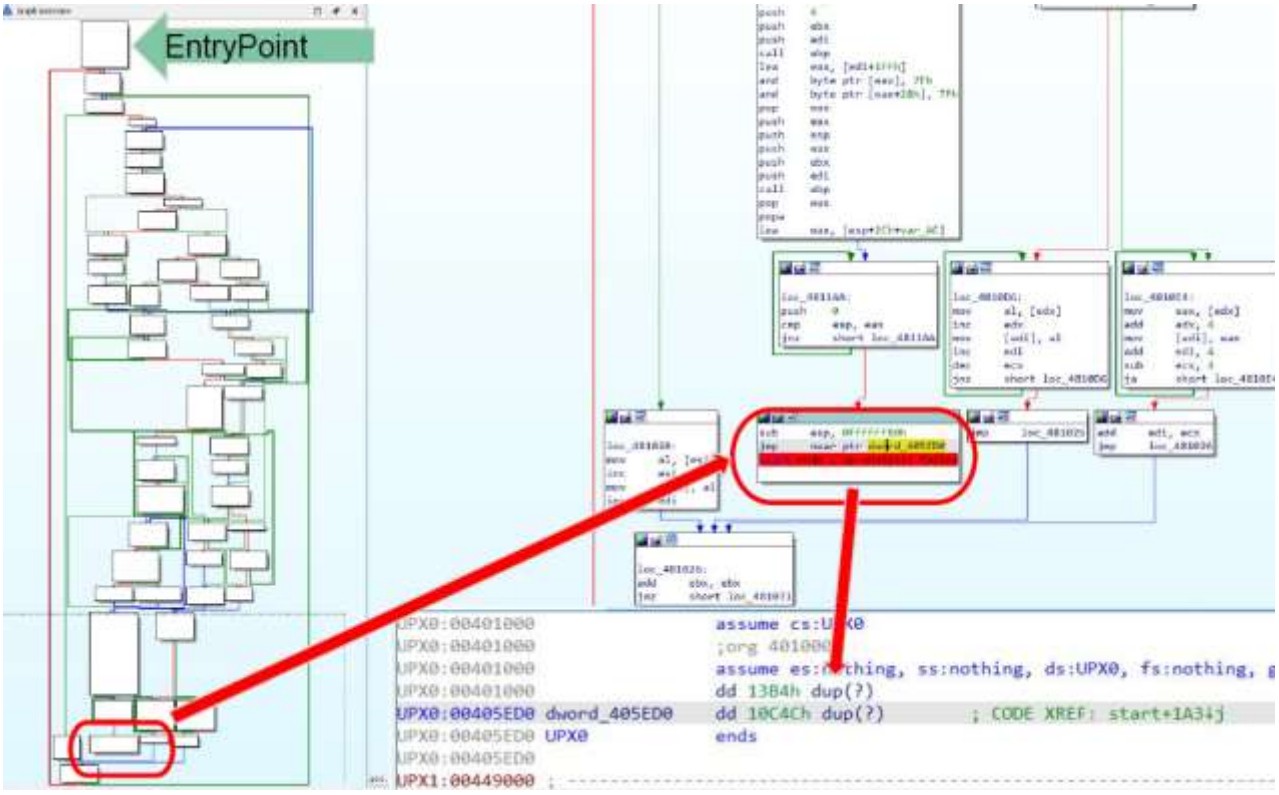
4. Atsidarome failą debugeriu (*IDA Pro* ir *Ghidra* instaliaciniai paketai turi įtrauktą vidinį debugerį. Šiuo atveju naudojamas atskiras „xdbg64“).

5. Sinchronizuojame disasemblerį ir debugerį pasinaudodami įskiepiu “ret-sync” (https://github.com/bootleg/ret-sync).

6. Disasembleryje randame „EntryPoint” įrašą, jį sufokusiuojame kodo analizatoriaus lange.

7. Jeigu atsidarę blokinės schemos vaizdą matoma neatpažintų segmentų, kurių dekompiliatorius negalėjo išardyti, gali būti jog failo dalis yra užšifruota.

8. Dalyje prieš neatpažintą segmentą (dažniausiai toliau seka užšifruota kodo dalis), ieškome „if” atitinkančios eilutės (pvz. CMP komanda). Didelė tikimybė, kad joje yra vykdomas patikrinimas dėl aplinkos/debugerio ir jų neaptikus įvykdomas atšifravimas.

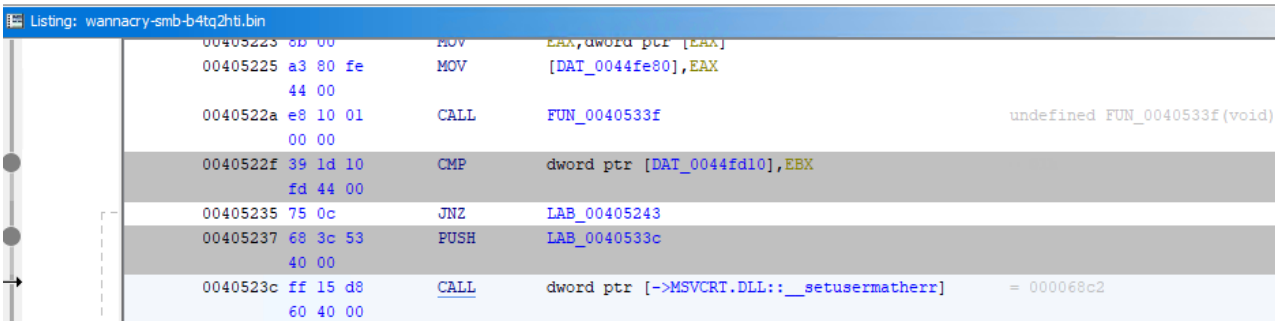


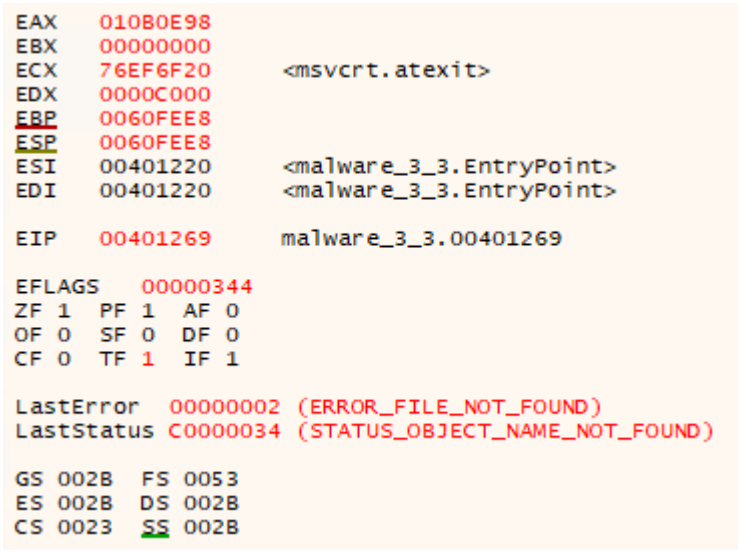
9. Įvedame sustabdymo tašką (pvz. „Ghidra sync“ įskiepyje - paspaudus F2). Sėkmingai sinchronizavus 5 žingsnyje, šis taškas turėtų atsirasti ir debugeryje. Stabdymas turėtų būti atliekamas prieš pat patikrinimo funkciją.

Tel. 1843
El. p. cert@nksc.lt
https://www.nksc.lt

10. Debugeryje paleidžiame KK vykdymą. Vykdymas automatiškai sustoja praeitame žingsnyje nustatytoje instrukcijoje.

11. Stebime steko reikšmes, kurias KK palygins prieš nutraukdamas procesą ar jį tęsdamas priklausomai nuo to ar buvo aptiktas debugeris ar smėliadėžė (angl. *sandbox*).





12. Pakeičiame tikrinamą reikšmę į mums naudingą ir tęsiame vykdymą po vieną instrukciją.

13. Pastebėję reikiamas iššifravimo instrukcijas ir įsitikinę, jog suradome teisingą tikrinimo vietą, galime sukurti modifikaciją (angl. *patch*) kenkėjiškam kodui, pakeisdami tikrinimo logiką į jmp instrukciją į mums reikalingą šifravimo funkciją (tokiu būdu apeidami patikrinimą).

14. Išsaugoję modifikuotą KK, jį importuojame į dekompiliatorių. Jeigu nėra antros šifravimo fazės, šįkart dekompiliatorius turėtų sėkmingai išanalizuoti visą failą.

Šioje vietoje atsiveria kenkėjiško kodo kompleksiškumas – dažnai netgi iššifravus toliau einančią logiką joje dar nėra aiškus tikrąsias funkcijas vykdantis kodas. Vietoje to jis gali bandyti susisiekti su nuotoline stotimi ir parsisiųsti sekančią dalį logikos, kuri vėlgi gali būti užšifruota keliais

Tel. 1843
El. p. cert@nksc.lt
https://www.nksc.lt

sluoksniais. Toks KK skaidymas etapais vadinamas „staging" (angl. *stage* - etapas). Iki galinės logikos, pvz. *backdoor*, *reverse-tcp* ir/ar *persistency* vykdymo tokių etapų gali būti nuo kelių iki keliolikos, tačiau dažnai greitai reakcijai saugant infrastruktūrą užtenka išnagrinėti pirmuosius kelis, siekiant išgauti tinkamus IOC panaudotinus užkardose ir skenavimuose.

Pavyzdys kaip Raspberry Robin kirminas naudoja trijų etapų struktūrą: https://thehackernews.com/2023/01/raspberry-robin-worm-evolves-to-attack.html

Komunikacijos

Pasinaudojant debugeriu galima palengvinti savo darbą analizuojant KK komunikacijas. Patogu uždėti sustabdymo taškus mus dominančiose funkcijose, pvz. prieš ir po jų vykdymą, kurios susijusios su tinklo veikla, pvz. *Socket*, *WNetAddConnection, NetApiBuffer* ir pan. Sėkmingai dešifravus KK turėtų būti matoma tiesiogiai funkcijų parametrai, t.y. eilutės (angl. *strings*) ar adresų nuorodos (*URLs*), kuriais bandoma susisiekti.

Pasinaudojant tinklo įrankiu, pvz. „WireShark" ar „tcpdump" lygiagrečiai galime pažingsniui tikrinti kiekvienos funkcijos atsakus ir jų paketų turinį.

Naudingi mokymosi resursai

NATO CCDCOE Atvirkštinės inžinerijos kursų knyga 2020

(https://ccdcoe.org/uploads/2020/07/Malware_Reverse_Engineering_Handbook.pdf)

NICCS CISA 101 Reverse Engineering Course (https://niccs.cisa.gov/education-training/catalog/federal-virtual-training-environment-fedvte/101-reverse-engineering)

REMnux tutorials (https://docs.remnux.org/)

Kuruojamas resursų sąrašas (https://github.com/albertzsigovits/malware-study)