KCKB UPJS

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH

Financované Európskou úniou NextGenerationEU

PLÁN OBNOVY

MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

8. Rozšírené techniky analýzy malvéru v prostredí Linux

Kontextuálna analýza a evolúcia hrozieb

- **Zmena paradigmy "Linux Security":**
  - Mýtus o "bezpečnosti vďaka minorite" je dávno prekonaný.
  - Linux poháňa 90 % verejného cloudu (AWS, Azure, GCP), superpočítače a kritickú IoT infraštruktúru.
  - Kompromitácia Linux servera má často dopad na tisíce koncových používateľov (supply chain attacks).
- **Špecifické ciele útokov:**
  - **Cloudová infraštruktúra a kontajnery (Kubernetes, Docker):** Útoky na orchestrátory a úniky z kontajnerov.
  - **Servery a databázy:** Ransomvér cielený na VMware ESXi hypervízory (hoci ESXi využíva vlastný VMkernel, útočníci využívajú jeho POSIX-kompatibilitu na spúšťanie Linux ELF malvéru).
  - **IoT a Embedded zariadenia:** Vytváranie masívnych botnetov (napr. Mirai) pre DDoS útoky.
- **Charakter a sofistikovanosť hrozieb:**
  - Dominancia **APT (Advanced Persistent Threats):** Štátom sponzorované skupiny (napr. Lazarus, Winnti) vyvíjajú Linux-native nástroje.
  - Posun boja z User-land do **Kernel-land:** Útočníci vedia, že EDR (Endpoint Detection and Response) riešenia monitorujú user-space API, preto sa presúvajú do jadra.
  - **"Living off the Land" (LotL):** Zneužívanie legitímnych nástrojov (ssh, curl, netcat, eBPF) na škodlivú činnosť.
- **Kľúčová výzva pre analytika:**
  - Tradičné postupy nefungujú.
  - Nutnosť chápať interné štruktúry jadra (task_struct, file operations), nie len systémové volania.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

Analýza shellkódu: Architektúra systémových volaní

- **Špecifiká Linuxu v porovnaní s Windows:**
  - Windows shellkód zvyčajne hľadá bázu `kernel32.dll` cez PEB (Process Environment Block) na vyriešenie API funkcií.
  - Linux shellkód interaguje priamo s jadrom cez **syscalls** (systémové volania), čím eliminuje závislosti na dynamických knižniciach (napr. libc). To ho robí extrémne kompaktným a nezávislým.
- **32-bit (x86/IA-32) konvencie:**
  - Mechanizmus: Vyvolanie softvérového prerušenia inštrukciou `int 0x80`.
  - Číslo syscallu: Uložené v registri `EAX`.
  - Argumenty (v poradí): `EBX`, `ECX`, `EDX`, `ESI`, `EDI`, `EBP`.
- **64-bit (x86-64) konvencie:**
  - Mechanizmus: Optimalizovaná inštrukcia `syscall` (opcode `0f 05`).
  - Číslo syscallu: Uložené v registri `RAX`.
  - **Kritický rozdiel v číslovaní:** Syscall čísla nie sú kompatibilné! (napr. `sys_write`: x86=4 vs x64=1; `sys_execve`: x86=11 vs x64=59).
  - Argumenty (v poradí): `RDI`, `RSI`, `RDX`, `R10`, `R8`, `R9`.
- **Polyglotný a hybridný kód:**
  - Malvér môže obsahovať vetvy pre obe architektúry.
  - Využitie techniky **"mode switching"**: 64-bitový proces môže prepnúť segmentový deskriptor (CS register) a vykonať 32-bitový kód, čo mätie debuggery a emulátory.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

Techniky "GetPC" a pozíciovo nezávislý kód (PIC)

- **Problém adresovania v moderných OS:**
  - **ASLR (Address Space Layout Randomization):** Bezpečnostná funkcia, ktorá náhodne umiestňuje zásobník, haldu a knižnice pri každom spustení.
  - Shellkód preto nemôže používať absolútne adresy (napr. `MOV EAX, 0x08048000`), pretože nevie, kde v pamäti sa práve nachádza.
- **Cieľ techniky GetPC (Get Program Counter):**
  - Získanie aktuálnej hodnoty Instruction Pointer (EIP/RIP) za behu.
  - Táto adresa slúži ako "kotva" (delta offset) pre výpočet relatívnych adries k dátam (napr. reťazec `/bin/sh` uložený na konci shellkódu).
- **Implementačné techniky:**
  - **Klasická metóda CALL/POP:**
    - `CALL` inštrukcia uloží adresu *nasledujúcej* inštrukcie na vrchol zásobníka.
    - Následný `POP` túto adresu vyberie do všeobecného registra.
  - **FPU inštrukcie (Floating Point Unit):**
    - Inštrukcie `FSTENV` alebo `FNSTENV` slúžia na uloženie stavu koprocesora do pamäte.
    - Súčasťou tohto stavu je aj adresa poslednej vykonanej FPU inštrukcie.
    - Postup: Vykoná sa triviálna FPU operácia (napr. `FLDZ`), následne `FNSTENV`, a z pamäte sa vyčíta EIP.
  - V modernom 64-bitovom prostredí (x86-64) techniky GetPC z veľkej časti strácajú zmysel, pretože architektúra natívne podporuje RIP-relatívne adresovanie (napr. LEA RAX, [RIP+0x50]), čo umožňuje priamy výpočet pozície bez nutnosti využívať CALL/POP alebo FPU inštrukcie.
- **Detekcia:**
  - Tieto sekvencie sú pre legitímny kód netypické.
  - Emulátory (libemu, Qiling) detegujú tzv. **self-reference behavior** – čítanie dát z vlastného inštrukčného toku.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

Polymorfizmus, Metamorfizmus a Obfuskácia

- **Polymorfný shellkód (Meniaci sa vzhľad, rovnaká funkcia):**
  - **Štruktúra:** Obsahuje zašifrovaný payload (telo) a unikátnu dešifrovaciu slučku (Decoder Stub).
  - **Mechanizmus:** Pri každej infekcii sa vygeneruje nový náhodný kľúč a nová verzia dekodéra. Payload zostáva po dešifrovaní rovnaký.
  - **Cieľ:** Obísť statické signatúry antivírusov a NIDS, ktoré hľadajú konkrétne bajtové sekvencie.
- **Metamorfizmus dekodéra (Prepisovanie kódu):**
  - Oveľa komplexnejšia technika, kde sa mení samotná logika a inštrukcie dekodéra, nie len šifrovanie.
  - **Zámena inštrukcií:** `MOV EAX, 0` môže byť nahradené za `XOR EAX, EAX`, `SUB EAX, EAX` alebo `AND EAX, 0`.
  - **Junk Code (Mŕtvy kód):** Vkladanie inštrukcií `NOP` alebo nezmyselných výpočtov, ktoré neovplyvňujú stav programu (napr. `ADD EBX, 0`), ale menia offsety a hash súboru.
  - **Transpozícia kódu:** Prehadzovanie poradia nezávislých inštrukcií a blokov kódu.
  - **Register Renaming:** Dynamická zmena používaných registrov v každej generácii.
- **Analýza a obrana:**
  - Statická analýza zlyháva (viditeľný je len unikátny, zakaždým iný dekodér).
  - Riešenie: **Emulácia CPU** (nástroje ako `sctest`, `Qiling`, `Unicorn Engine`).
  - Analytik nechá kód bežať v bezpečnom, emulovanom prostredí. Počká, kým dekodér vykoná svoju prácu a deteguje moment, kedy riadenie prejde do dešifrovanej oblasti pamäte.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

Analýza šifrovaných vzoriek

- **Shannonova entropia v kontexte malvéru:**
  - Matematická miera náhodnosti alebo neusporiadanosti dát v súbore.
  - Rozsah hodnôt: 0 (samé nuly) až 8 (dokonale náhodné dáta) bitov na bajt.
- **Indikátory pre analytika:**
  - **Nízka entropia (< 6.0):** Bežný strojový kód, textové reťazce, sekvencie nulových bajtov (padding).
  - **Vysoká entropia (> 7.5):** Silný indikátor kompresie alebo šifrovania. Kvalitne zašifrované dáta sú nerozoznateľné od náhodného šumu.
- **Praktická aplikácia a nástroje:**
  - **Nástroj `binwalk`:** Dokáže vygenerovať graf entropie pre celý súbor.
  - **Analýza ELF sekcií:** Ak sekcia `.text` (kde má byť spustiteľný kód) má entropiu 7.9, je takmer isté, že súbor je "zabalený" (packed).
  - Binwalk tiež slúži na extrakciu vnorených súborových systémov alebo pripojených payloadov (overlay data).
  - Ďalšie nástroje: Detect It Easy, radare2/rizin
- **Význam pre proces analýzy:**
  - Vysoká entropia je signálom "STOP" pre statickú analýzu. Príkaz `strings` nevráti nič čitateľné.
  - Analytik musí najprv pristúpiť k rozbaleniu (unpacking) alebo dešifrovaniu.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

Manuálne rozbaľovanie (Unpacking): prípadová štúdia UPX (32-bit)

- **Problém s automatizáciou:**
  - UPX je najčastejší packer, ale malvér takmer vždy modifikuje hlavičky (napr. zmení magic bytes "UPX!" na "YXZ!"), čím znefunkční štandardný príkaz `upx -d`.
  - Preto je nutné manuálne rozbaľovanie.
- **Generická metodológia v debuggeri GDB:**
  - **Hľadanie OEP (Original Entry Point):** Cieľom je nájsť adresu, kde packer odovzdá riadenie pôvodnému, rozbalenému kódu.
  - **Sledovanie inštrukcie `POPAD` / `POPA`:** Väčšina packerov (vrátane UPX) na začiatku uloží všetky registre (`PUSHAD`) a tesne pred koncom ich obnoví (`POPAD`).
  - **Technika Hardware Watchpoints:**
    - Po štarte programu (`starti`) zistíme adresu zásobníka (`ESP`).
    - Nastavíme `watch *0xAdresaESP` (sledovanie zápisu/čítania na zásobníku, kde sú uložené pôvodné registre).
    - Keď packer vykoná `POPAD`, debugger sa zastaví tesne pred skokom na OEP.
  - **Memory Dump a Fixup:**
    - Po skoku na OEP je proces v pamäti rozbalený.
    - Použijeme GDB príkaz `dump memory` na uloženie obrazu procesu.
    - **Kritický krok:** Oprava (Reconstruction) ELF hlavičiek a tabuliek symbolov, aby nástroje ako IDA Pro alebo Ghidra vedeli súbor správne načítať.

Financované Európskou úniou
NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

Dynamická inštrumentácia (Frida): Obchádzanie šifrovania

Využitie pri analýze malvéru:

- Hookovanie (zachytávanie) funkcií, sledovanie parametrov a návratových hodnôt.
- Modifikácia logiky aplikácie za behu (napr. obídenie volania ptrace(PTRACE_TRACEME) alebo podvrhnutie odpovede pri čítaní súboru /proc/self/status, aby malvér nevidel, že je debugovaný).
- **Prípadová štúdia: Dešifrovanie SSL/TLS prevádzky:**
  - Moderný malvér komunikuje cez HTTPS.
  - Namiesto zložitej kryptoanalýzy šifrovania alebo detekcie inštalácie CA certifikátov použijeme Fridu.
  - **Postup:** Hookneme funkcie `SSL_write` (kde sú dáta ešte v plaintexte pred odoslaním) a `SSL_read` (kde sú dáta už dešifrované po prijatí).
  - Týmto spôsobom "odpočúvame" komunikáciu priamo v pamäti procesu, ešte predtým, než ju spracuje SSL knižnica.
  - Pri staticky linkovaných a stripped binárkach (typické pre malvér v jazykoch Go/Rust) nie je možné priamo hooknuť funkciu SSL_write podľa názvu. Analytik musí najprv reverzným inžinierstvom zistiť presnú pamäťovú adresu (offset) šifrovacej funkcie a Fride odovzdať tento offset.
- **Výhoda:**
  - Funguje aj pri staticky linkovaných knižniciach.
  - Obchádza techniku **Certificate Pinning**, ktorá by inak zabránila použitiu proxy (napr. Burp Suite).

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

XOR Dekódovanie a automatizácia analýzy

XOR (Exkluzívny logický súčet):

- Najčastejšia metóda "šifrovania" v malvéri pre svoju rýchlosť a jednoduchosť.
- Kľúčová vlastnosť: Je to reverzibilná operácia (A xor B xor B = A). Rovnaká funkcia šifruje aj dešifruje.

Prípadová štúdia: XorDDoS:

- Linuxový botnet, ktorý používa rotujúce XOR kľúče na ochranu svojich konfiguračných dát a zoznamov C2 serverov.
- Často používa viacvrstvové šifrovanie (napr. kompresia + XOR).

Analytický postup:

- **Identifikácia:** Nájdenie slučky v disassembleri, ktorá načíta bajt z pamäte, vykoná operáciu `XOR` s kľúčom a výsledok uloží späť.
- **Extrakcia kľúča:** Kľúč môže byť statický (jeden bajt), reťazec (multibyte key) alebo dynamicky generovaný.

Nástroje na kryptoanalýzu:

- `xortool`: Nástroj na frekvenčnú analýzu. Funguje na princípe, že v binárnych súboroch je najčastejším bajtom `0x00` (NULL). Ak zašifrujeme nulu XORom (`0x00 ^ KEY`), výsledkom je samotný kľúč (`KEY`). `xortool` hľadá najčastejší bajt a predpokladá, že je to kľúč.

Financované Európskou úniou
NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

Identifikácia C2: Domain Generation Algorithms (DGA)

Motivácia:

- Útočníci sa chcú vyhnúť tzv. blacklistingu (zablokovaniu) statických domén alebo IP adries.
- Riešením je generovať tisíce domén denne, pričom útočník zaregistruje len jednu z nich.

Typológia DGA algoritmov:

1. **Time-based (Závislé na čase):**
   - Seed (zárodok) pre generátor je odvodený z aktuálneho dátumu/času.
   - Príklady: CryptoLocker, niektoré vezie Kinsing
   - Výhoda: Synchronizácia bez komunikácie (bot aj C2 server vedia, aký je dátum).
2. **Seed-based (Deterministické):**
   - Generovanie z pevného číselného seedu alebo kurzových lístkov, počasia, atď.
3. **Dictionary-based (Slovníkové):**
   - Spájanie náhodných slov zo slovníka (napr. `blue-carpet-sky.com`).
   - Cieľ: vyzerať ako legitímna doména a obísť ML detekciu.
4. **preAnalýza a protiopatrenia:**
   - **Reverzné inžinierstvo:** Analytik musí nájsť PRNG (Pseudo-Random Number Generator) funkciu v kóde malvéru a zistiť, ako sa tvorí seed.
   - **Predikcia:** Po pochopení algoritmu môže analytik vygenerovať zoznam domén na nasledujúce obdobie a proaktívne ich zablokovať (DNS Sinkhole).
   - **Detekcia pomocou ML:** Neurónové siete sú na rozlišovanie medzi "ľudskými" doménami a náhodnými zhlukmi znakov (napr. `xyza78bbb.com`).

*Poznámka: V prípade moderného Linux/IoT malvéru sa útočníci namiesto DGA často presúvajú k **P2P sieťam** (napr. botnety Hajime, Mozi).*

.

Financované Európskou úniou
NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

Extrakcia kľúčov z Heap

- **Problém:** Ransomvér a botnety musia mať dešifrovacie kľúče (AES, ChaCha20) v RAM.
- **Heap Analysis:** Hľadanie charakteristických štruktúr **AES Key Schedule**.
- **Nástroje:** `aeskeyfind`, `findaes`.
- **Výsledok:**
  - Možnosť dešifrovať súbory ransomvéru (ak beží proces).
  - Dešifrovanie zachytenej sieťovej komunikácie (PCAP).
- *Poznámka: Pred použitím týchto nástrojov musí analytik vykonať dump pamäte. V Linuxe to možno urobiť na úrovni procesu cez vytvorenie coredumpu (napr. nástrojom gcore), z pamäťového priestoru /proc/<pid>/mem, alebo na úrovni celého stroja vytvorením forenzného obrazu RAM pomocou modulu jadra LiME.*

Financované Európskou úniou
NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

Záver

- **Zhrnutie:** Boj sa presúva do jadra (eBPF, BPFDoor).
- **Nutné zručnosti:**
  - Nízkoúrovňové znalosti Linuxu (syscalls, memory management).
  - Kombinácia statickej (IDA/Ghidra) a dynamickej (Frida, emulácia) analýzy.
  - Forenzná analýza pamäte ako štandard.
- **Budúcnosť:** Detekcia anomálií v eBPF a využitie AI na deobfuskáciu.

Financované Európskou úniou
NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

KCKB UPJS

UNIVERZITA
PAVLA JOZEFA ŠAFÁRIKA
V KOŠICIACH

Financované
Európskou úniou
NextGenerationEU

PLÁN OBNOVY

MINISTERSTVO
INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA
A INFORMATIZÁCIE
SLOVENSKEJ REPUBLIKY

Ďakujem za pozornosť.