Implementace systémů HIPS: historie a současnost

Martin Dráb
martin.drab@secit.sk

HIPS: základní definice

- Majoritně používané operační systémy disponují bezpečnostními modely, které dovolují jednotlivým uživatelům určit, co smějí a co nesmějí.
- V případě Windows je většina domácích uživatelů zvyklá pracovat neustále s oprávněním administrátora, což činí bezpečnostní model značně neefektivní.
- Systémy HIPS – programy, které monitorují, oznamují (a blokují) podezřelé aktivity v operačním systému.

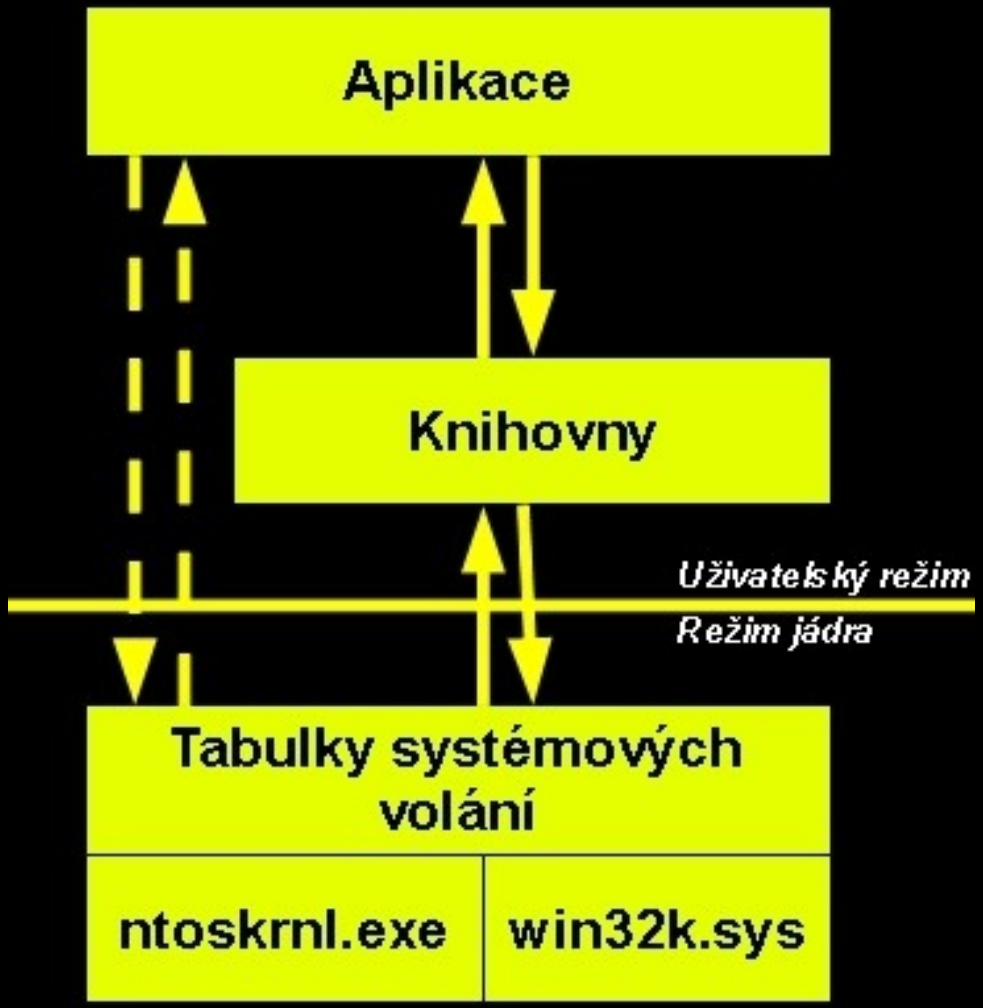
Čemu se budeme věnovat

- Různým technikám použitelným při implementaci systémů HIPS a jejich problůmům
- Rozhraním, která pro ulehčení (a umožnění) implementace poskytuje Microsoft.
- Složitostí realizace ochrany určitého aspektu systému (souborového systému, registru, procesů, GUI...)

Na které techniky se zaměříme

- Modifikace kódu (neomezený dosah)
- Modifikace tabulek systémových volání (neomezený dosah)
- Direct Kernel Object Hooking (objekty jádra)
- OB Filtering Model (procesy, vlákna)
- Monitorování a ochrana registru
- Windows Filtering Platform (síť)

Architektura systému



Modifikace kódu

- Součást systému HIPS pozmění kód rutin zajišťujících operace, které chce sysystém monitorovat, popř. Blokovat (přístup k procesům, změny nastavení OS...).
- Cílová rutina je obvykle upravena tak, že se při jejím zavolání stane následující:
  - Volání je přesměrováno do modulu systému HIPS
  - Pokud se HIPS rozhodne, že operaci blokovat nebude, je vykonán kód původní rutiny.
  - Před předáním řízení volajícímu může proběhnout úprava výsledků operace.

Modifikace kódu

- **Výhody:**
  - Lze provádět prakticky kdekoliv (v knihovnách, v jádře, přímo v aplikaci)
  - Prakticky neomezená oblast působnosti
- **Nevýhody:**
  - Závislé na architektuře procesoru
  - Přepisuje se cizí kód, což není vždy dovoleno
  - Možný zdroj nestability (zvláště když se více entit pokusí modifikovat stejnou funkci)

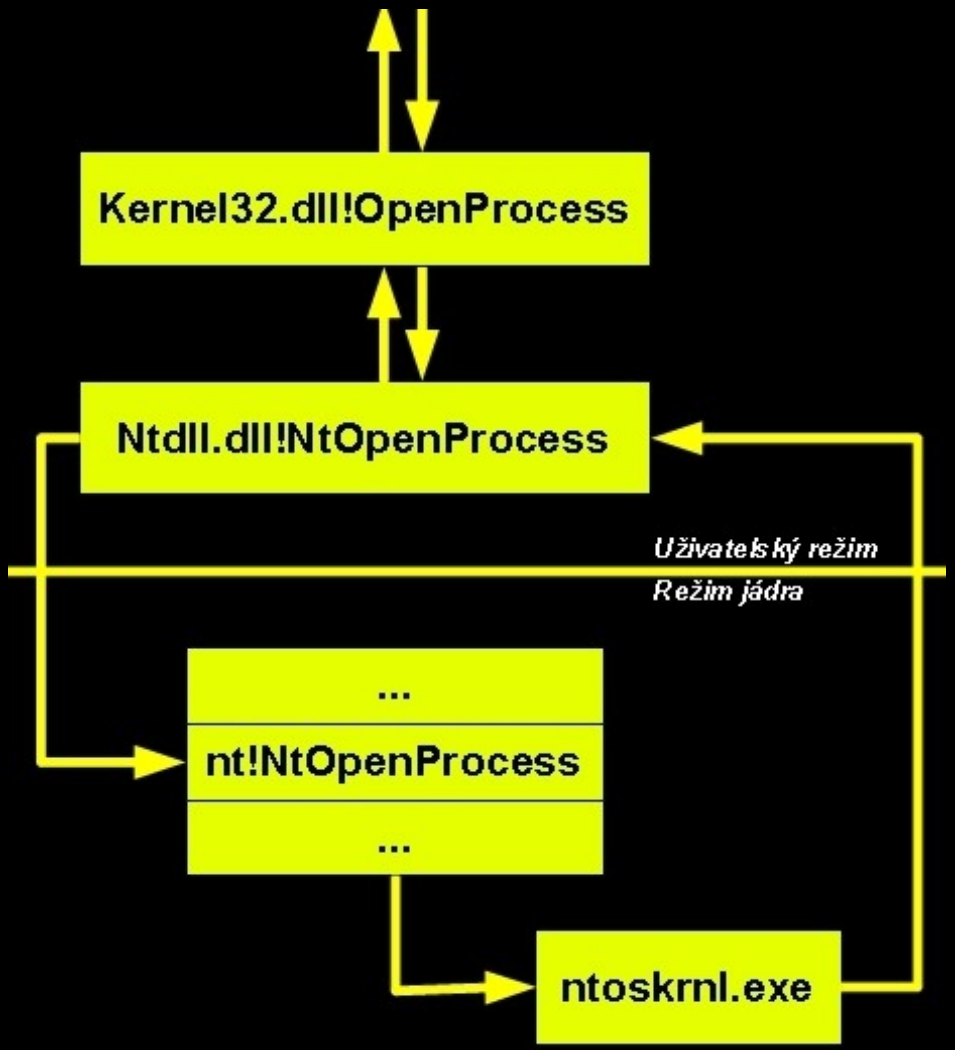
Systémová volání

- Kód aplikací (a jimi používaných knihoven) je vykonáván v uživatelském režimu procesoru
- Pro některé operace je potřeba mít vyšší oprávnění
- O provedení takových operací musí aplikace požádat jádro operačního systému. Vyslání požadavku = **systémové volání**
- Jádro může danou operaci provést, ale může také odmítnout (např. Kvůli bezpečnostnímu modelu).

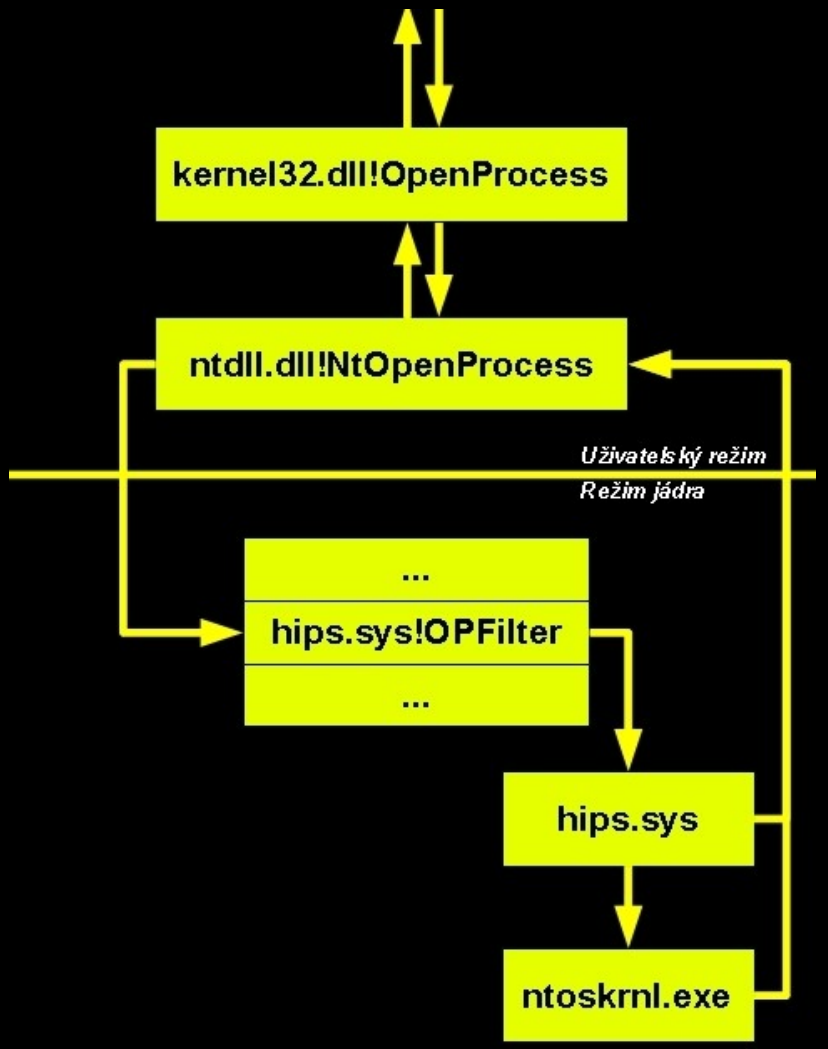
Průběh systémového volání

- Aplikace zavolá knihovní funkci
- Knihovna (může jich být i více) překonvertuje volání dané funkce na požadavek pro jádro
- Vykoná se speciální instrukce pro přechod do režimu jádra
- Operační systém předá řízení rutině uvedené na daném místě v tabulce systémových volání
- Rutina se pokusí zkopírovat obsah argumentů sytémového volání do režimu jádra. Následně se pokusí provést zadanou operaci

Průběh systémového volání



Modifikace tabulky systémových volání



Problémy při modifikaci tabulek systémových volání

- Rutina systému HIPS převezme řízení před příslušnou funkcí jádro operačního systému.
- Obsah argumentů systémového volání ještě není překopírován do paměti jádra
- Rutina systému HIPS tedy musí provádět vlastní ověřování platnosti argumentů a kopírovat jejich obsah do paměti jádra, aby s nimi mohla bezpečně pracovat
- Pokud HIPS příslušnou operaci povolí, jeho rutina předá řízení funkci původně zapsané v tabulce systémových volání

Problémy při modifikaci tabulek systémových volání

- Původní rutina „neví“ o tom, že se již obsah argumentů volání nachází v paměti jádra a znovu jej kopíruje z uživatelského režimu
- Dochází tedy k dvojímu kopírování obsahu argumentů z paměti uživatelského režimu.
- Mezi těmito dvěma operacemi se ale obsah argumentů může změnit (jde o paměť přístupnou aplikaci)!
- Tento problém se v minulosti objevil již několikrát a pod různými jmény: TOCTOU, argument switch attack, KHOBE...

KHOBE

- V r. 2010 existoval u mnoha bezpečnostních balíků, aktuální situaci neznám
- Objevuje se hlavně na 32bitových verzích Windows, kde se modifikace kódu a tabulek systémových volání hodně používaly
- Není mi známo, že by byl zneužit ITW
- S nástupem 64bitových Windows zmizí, ačkoliv možná se dočkáme znovuzrození

Modifikace tabulek systémových volání

- **Výhody:**
  - Dovoluje kontrolovat skoro každý aspekt života aplikací
  - (zdánlivě) jednoduchá implementace
- **Nevýhody:**
  - Závislé na konkrétní verzi Windows
  - Závislé na architektuře procesoru
  - Ověřování obsahu argumentů, KHOBE
- **Praxe:** na 32bitových Windows téměř každý bezpečnostní balík

Direct Kernel Object Hooking (DKOH)

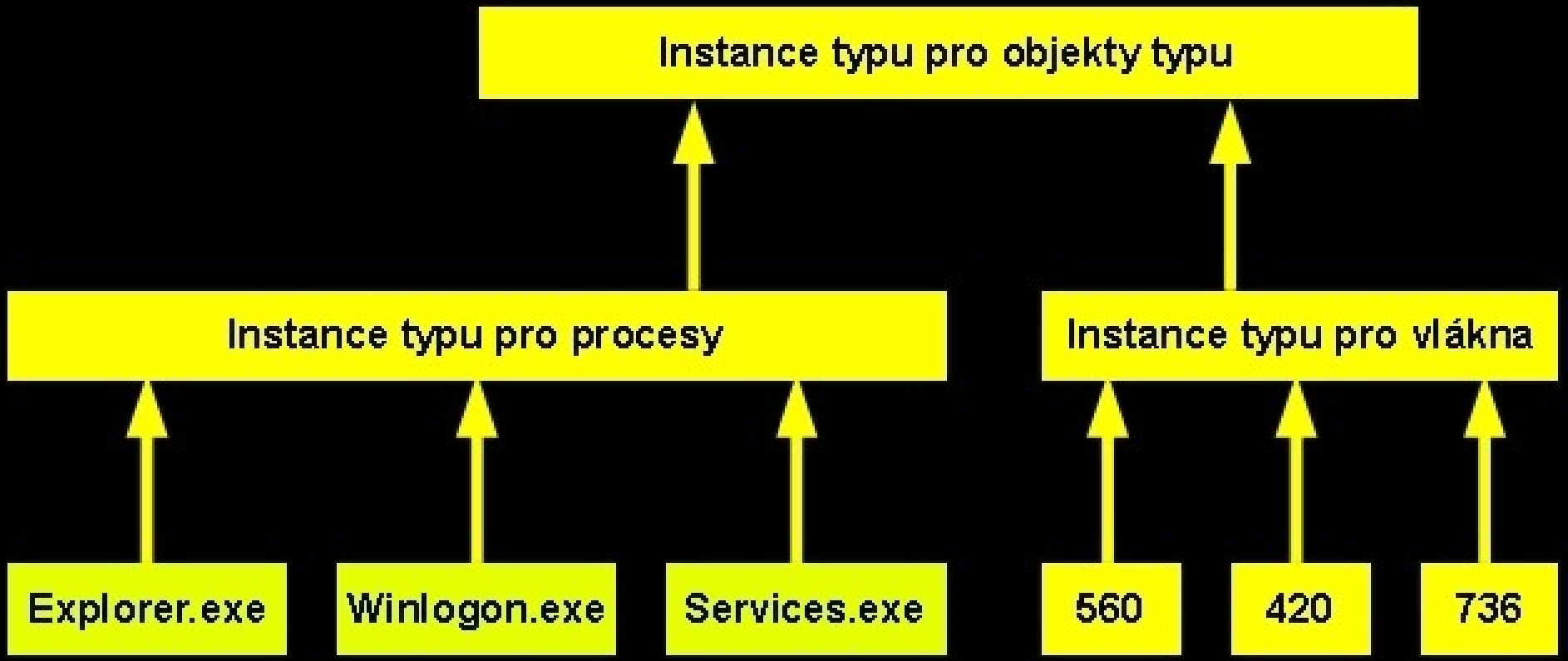
- Jádro Windows reprezentuje mnoho entit (procesy, vlákna, klíče registru, otevřené soubory...) jednotným způsobem; entitou zvanou **objekt jádra** (kernel object)
- Jednotná reprezentace dává těmto entitám například možnost pojmenování či nastavení ochrany v rámci bezpečnostního modelu.
- Aplikace s objekty jádra pracují prostřednictvím nepřímých odkazů – **handle**. Každé handle v sobě nese zakódovanou informaci o cílovém objektu a o tom, co skrz něj může s objektem aplikace dělat.

Direct Kernel Object Hooking (DKOH)

Každý objekt jádra ve své struktuře má i odkaz na tzv. **objekt typu**, který udává základní charakteristiky pro daný druh entit (např. Procesy).

- Součástí každého objektu typu jsou adresy několika rutin, které jádro vykoná při určitých událostech. Například:
  - Při pokusu o vytvoření nového handle
  - Při rušení existujícího handle
  - Při odstraňování objektu z paměti
  - Při získávání jména objektu

Direct Kernel Object Hooking (DKOH)



Direct Kernel Object Hooking (DKOH)

- Některé z těchto rutin lze využít pouze pro monitorovací účely.
- Jiné (například tu, vykonávanou při vytváření nového handle) lze využít i k blokování příslušných operací.
- Použitím této techniky tak systém HIPS má kontrolu například nad přístupy k objektům jádra určitého typu.

Direct Kernel Object Hooking (DKOH)

- **Výhody:**
  - Imunní proti KHOBE
  - Není nutné kopírovat data z uživatelského režimu
- **Nevýhody:**
  - Definice rutin se často mění
  - Čekání na odpověď uživatele je problematické
  - Nelze použít na 64bitových systémech
- **Praxe:** SandboxIE, Rootkit Unhooker

OB Filtering Model

- „nadstavba“ nad DKOH přímo od Microsoftu
- Funguje i na 64bitových verzích Windows
- Dovoluje monitorovat (někdy i blokovat či měnit) přístup k objektům jádra
- Zatím podporovány pouze procesy a vlákna
- Dává jistou kontrolu pouze nad vytvářením handle, ostatní operace nejsou podporovány.
- Dostupné od Windows Vista SP1

OB Filtering Model

- **Výhody:**
  - Funguje i na 64bit systémech
  - Legální
  - Teoretická možnost „ručního“ rozšíření i na další druhy objektů
- **Nevýhody:**
  - Podporuje pouze procesy a vlákna
  - Dovoluje kontrolovat pouze tvorbu handle
  - Ne vždy lze přístup blokovat

Ochrana registru

- Dříve nebyla jiná možnost než modifikace tabulek systémových volání
- Ve Windows XP se objevilo rozhraní, které dovolovalo ovladačům monitorovat či blokovat některé operace
- Rozhraní postupně vylepšováno v následujících verzích operačního systému a v rámci aktualizací Service Pack
  - Podpora dalších operací s registry
  - Větší možnost ovlivňování výsledků operací
  - Pohodlnější implementace

Ochrana registru

- Princip: ovladač zaregistruje jednu rutinu, kterou jádro volá při každé operaci nad registrem.
- Rutina může operaci blokovat vrácením hodnoty indikující chybu.
- Windows Server 2003 tento koncept rozšiřuje tak, že každá úspěšná operace nad registrem znamená dvě volání zaregistrovaných rutin.
  - **Pre notifikace:** informuje o tom, co se bude dít
  - **Post notifikace:** informuje o tom, co proběhlo

Ochrana registru

- **Výhody:**
  - Relativně jednoduché použití, zvláště pro novější verze OS (Vista, 7)
  - Dovoluje nejen blokování, ale i změnu výsledků operace.
  - Použitelné i na 64bitových systémech
- **Nevýhody:**
  - Implementace kompatibilní se všemi verzemi rozhraní není triviální