O C H R A N A D A T V I N T E R A K T I V N Í C H P R O S T Ř E D Í C H

Z. Rusín

Úroveň a komplexnost ochrany uživatelských dat je ve světě jedním z důležitých kritérií uživatelského hodnocení výpočetních systémů. V domácích podmínkách trvale neuspokojené poptávky po výpočetní technice bývá problematika ochrany uživatelských dat při pořizování výpočetních systémů opomíjená a její naléhavost je zjišťována až po zavedení prvých interaktivních úloh do rutinního provozu.

Dlouholetou celosvětovou praxí uživateli identifikované a dodavateli programového vybavení na různé úrovni implementované základní tři okruhy ochrany dat v dávkovém prostředí, jimiž jsou: přidělování vlastnických a přístupových práv k souborům a jejich magnetickým nosičům, na vlastnických právech založený systém bezpečnostních a archivních kopií souborů a ochrany adresního prostoru uživatelské úlohy, vůbec neřeší stěžejní otázku interaktivních aplikací, kterou je verifikace oprávněnosti konkrétní osoby ke konkrétní transakci v konkrétní aplikaci v daném čase a obecně z libovolného koncového zařízení.

Absolutistické řešení, vyžadující po koncovém uživateli osobní identifikaci při každé interakci je téměř vždy nepřijatelné, vynucuje-li jakoukoli uživatelovu aktivitu. Visuální identifikaci osoby či snímání otisku prstu z klávesnice bohužel všichni asi zařadíme mezi rekvisity sci-fi literatury.

Prakticky vždy je nutno se uchýlit k systému osobních hesel ověřovaných při vstupu do aplikace a po každém nepřiměřeně dlouhém přerušení toku interakcí. I tento primitivní způsob má však smysl jen tehdy, umožňuje-li interaktivní zařízení potlačit viditelnost hesla ve tvaru netriviální posloupnosti znaků, je-li systém hesel vybudován tak, aby heslo bylo jedinečné a nebylo zjistitelné ani v aplikačním programu ani v diagnostice.

Naskýtá se námitka, že i pak může konkrétní osoba své heslo libovolně často prozradit. Jistě. Spojí-li se však s identifikací

evidovanou aplikačním programovým vybavením systém skutečné hmotné odpovědnosti za prováděné operace, minimalisujeme snad i toto nebezpečí.

V opačném případě jde o programátorsky i počítačově nákladný formalismus budící u koncového uživatele pocit znechucení tím větší, čím větší je jeho závislost na aplikaci. Rozvíjení úvah v tomto směru se vymyká účelu článku; nicméně je autorovi nevesele z toho, že předchozí konstatování vzhledem ke skutečnému stavu uplatňování hmotné odpovědnosti v naší ekonomice odsoudí dále popisovanou možnost implementace osobních hesel pro mnoho čtenářů do sféry utopií.

Identifikace heslem proti instanci uživatele či účtu je v interaktivních prostředích běžná. Pojmy "uživatel", "účet", převzaté z dávkového prostředí, však málokdy označují individuální osobu. Jejich reálnou funkcí je poměrně vágní označení subsystému, aplikace, programátorského týmu, specialisované činnosti nebo vlastníka souborů a medií.

Systémy s hesly vztaženými k datovým souborům se pro identifikaci osoby koncového uživatele už vůbec nehodí.

Téměř každé aplikace v našem smyslu je strukturovaná a ne každá osoba bude esi oprávněné a způsobilá ke všem činnostem v rámci aplikace. S osobním heslem musí být spřažena datové struktura popisující přístupová práva dané osoby k činnostem aplikací realisovaným. Patrně postačí jednoduchá bitová maska, jejíž neměnnou interpretaci provede programové vybavení. Je zřejmé, že pak lze v uživatelských programech realisovat dynamickou "menu" strukturu, kde nabídka je odvozena od osobních přístupových práv koncového uživatele k aplikaci. Privilegovanou činností je řízení aplikace z hlediska přístupových práv zúčastněných osob.

Takto chápaný pojem aplikace umožňuje do činností v rámci aplikace zahrnout všechny akce uživatelské, programovací, organizační i projekční a využívat jednotného systému evidence a hmotné odpovědnosti po celou dobu životnosti aplikace (ve smyslu práce /1/).

Stejným způsobem lze s rozumným rozsahem programovacích prací organisovat i všechny činnosti v rámci řízení, údržby a rozvoje samotného výpočetního systému, jež je vždy nutno charakterisovat jako privilegované už proto, že rizika chybných zásahů přesahují

rozměr jedné aplikace. Je dokonce zcela na místě požadovat, aby právě u těchto činností byl systém osobní identifikace zaveden nejdříve (jeho implementátoři jej vztáhnou na sebe), což mimo jiné určitě zvýší pravděpodobnost jeho realizace v uživatelském prostředí.

Úvahy o způsobu implementace systému osobních hesel zahajme rekapitulací obvyklých způsobů ochrany datových souborů. Zjistíme, že účinný mechanismus privatisace souborů a medií, který je nedílnou součástí operačního systému, je nezbytný jak pro komplexní řešení ochrany adresního prostoru jednotlivé úlohy, tak pro automatisovaný systém archivních a bezpečnostních kopií souborů.

Bylo by podivné, kdyby osobní identifikace nebyla rozumně realisovatelná pomocí týchž principů.

V každém výpočetním systému s magnetickými medii nalezneme adresář souborů. Překročí-li jeho implementace triviální funkci kartotéky registrující obsah magnetického nosiče dat, máme vždy do činění s datovou strukturou reflektující vztahy mezi soubory, jejich zřizovateli a jejich nosiči. Stačí každému existujícímu vztahu přisoudit hodnotu určující přístupové práva jednoho objektu k druhému a od ní odvodit automatickou kontrolu oprávněnosti přístupu k objektu ve chvíli jeho přiřazování v konkrétním procesu, jehož zřizovatelem je konkrétní vlastník representovaný instancí uživatele nebo účtu v dříve zmíněném smyslu. V tu chvíli se stává adresář souborů centrální sdílenou datovou strukturou v daném operačním systému, může být implementován obecně známými databázovými prostředky a jeho funkci lze zobecnit až na universální nástroj parametrisace všech potenciálně proměnných hodnot určujících chování výpočetního systému. V tomto směru má čtenář k dispozici autorovy texty /2/ až /5/. Místo adresář souborů říkejme této databázi systémový katalog.

Je-li tento model privatisace objektů implementátory operačního systému přijat, nebrání nic zobecnění katalogové informace o objektu natolik, aby například pro datový soubor obsahoval popis jeho fyzického uspořádání a umístění, ale též předpis pro automatický mechanismus záložních kopií a rovněž všeobecná přístupová práva

vztažená k vícestupňovému uspořádání privilegovanosti vlastníků a též k vícestupňovému systému privilegovanosti kódu v jednom procesu.

Katalog pak sám sebe deklaruje jako soubor všeobecně přístupný z privilegovaného kódu, přičemž různé údaje o tomtéž objektu jsou pro čtení a zápis obecně přístupné z různých úrovní privilegovanosti.

Archivní kopie souboru může být pak považována za datový soubor v témž smyslu co soubor původní, snad s jiným popisem, ale s výhodou téhož mechanismu přístupových práv.

Frontu požadavků na archivaci či kopírování, obsluhovanou vhodným typem systémových procesů, lze implementovat jako zvláštní typ vztahů v rámci katalogu, jež se po úspěšném provedení kopie transformují v relaci mezi souborem a jeho kopií. Jedinou nevýhodou je tendence k dlouhodobému narůstání katalogu. Alternativou je implementace archivního systému s tradičním adresářem souborů a interním formátem dat, kde soubor může být ze své kopie restaurován výhradně v rámci archivního systému. První řešení nic podobného neimplikuje, naopak lze zachovat totožnost souboru a jeho kopie na úrovni fysických bloků dat.

Ani v druhé variantě asi neopustíme myšlenku spolingu - asynchronnost požadavku a jeho realizace, ať už periodické nebo podle jiných provozních kriterií. Výhodou katalogového popisu je jak soustředění parametrické informace, tak její variabilita. Takové údaje jako expirační doba kopie, počet uchovávaných versí, kdy vůbec kopii provádět (např. po každé aktualisaci), musí být jinak parametry těch systémových prostředků, jimiž požadavky na provedení kopie zařazujeme do fronty výkonnému procesu, přičemž nelze docílit automatismu.

Implementátoři katalogu většinou využití i možnosti omezit velikost katalogu tím, jak definují pojem knihovny. Katalog pak obsahuje popis knihovního indexu, vzorový popis knihovního souboru povinný pro všechny knihovní soubory a jediná přístupová práva platná pro celou knihovnu. Index je tedy opět adresářem souborů v původním smyslu a měl by obsahovat dostatečné údaje pro snadnou implementaci archivních kopií knihovních souborů. Tento požadavek je důležitý pro katalogový mechanismus kopií souborů. Má-li systém kopií vlastní

adresář, lze v něm pomocí jmen snadno rozlišovat mezi samostatným a knihovním souborem.

Existence knihoven v tomto pojetí komplikuje i jinou, dosud nediskutovanou otázku - katalogování obsahu cizího magnetického media. Buďto je na disku uchováváno dostatek informací o jeho obsahu, čímž se vracíme k tradičním adresářům, nebo lze při exportu media vytvořit zvláštní soubor s příslušnou kopií katalogových informací o mediu, souborech i jiných vlastnících včetně přístupových práv platných na exportující instalaci, který je na disku indikován třeba v návěští svazku, nebo v krajním případě privilegovanou katalogovou funkcí zaváděn na importující instalaci manuálně podle průvodní dokumentace. Tato praxe je běžná při distribuci software. V závislosti na interním formátu disku lze konstruovat metody "ošahávání" neznámého media. Naznačené obtíže jsou však vždy akceptovatelné, neboť se vztahují k neběžným privilegovaným operacím, přičemž úroveň ochrany uživatelských souborů je podstatně vyšší než v systémech s tradičními adresáři (je např. vyloučeno přepsat některému vlastníku obsah magnetické pásky chybnou manipulací).

Poněkud odlišnou záležitostí je ochrana adresního prostoru jednotlivého procesu v rámci stránkovacího mechanismu operačního systému. Komplexní řešení zde zasahuje až k hardwarovým principům daného výpočetního systému. Z našeho hlediska je podstatné, aby zabraňovalo absolutnímu adresování paměti v uživatelských procesech, potlačilo jakýkoli pokus o přístup k sekundárním stránkovacím diskovým prostorům mimo mechanismus stránkování a zamezilo nechtěnému výskytu soukromých a privilegovaných dat v diagnostických výpisech.

První požadavek je splnitelný tím, že cílový modul je produkován výhradně v relativní formě (viz /4/), což implikuje transformaci relativních adres alespoň na úrovni mikrokódu procesorů. K druhému požadavku lze využít katalogovaných přístupových práv pro přístupnost pouze z dostatečně privilegovaného kódu v rámci všech procesů. Třetí požadavek je v rozporu s potřebami diagnostiky, neboť právě utajovaná data mohou být příčinou havárií. Nicméně diagnostiku lze strukturovat a privilegovanou část provádět z procesů ovládaných proponovaným systémem osobní identifikace. Snímky reálně

paměti na magnetických mediích lze zabezpečit opět vhodnými katalogovanými přístupy.

Každá datová oblast ve virtuálním paměťovém prostoru přísluší jedinému procesu, jímž je zřizována s potřebnými vlastnostmi - globalitou, stupněm potřebné privilegovanosti kódu, utajeností obsahu. Utajenost je zabezpečována přemapováním reálné paměti při každém stránkovacím přesunu a může být respektována i diagnostickými systémy včetně monitorování komunikačních linek. Kódování informace mezi modemy by mělo zabezpečit i přenosové cesty.

Pozorný čtenář v průběhu výkladu jistě objevil skrytou závislost našeho pojetí privilegované funkce v rámci aplikace na viditelnosti či přístupnosti jí realisujících programových prostředků jen z kódu samotné aplikace a to ještě v závislosti na přístupových právech konkrétní osoby. Teoreticky je to pouze problém vhodného strukturování knihoven cílových modulů, kdy moduly z různých knihoven si předávají data technikou fortranovských common oblastí. Této podmínce je možno vyhovět vyvíjíme-li programy s vědomím potřeby dodržet tuto zásadu. Nic podobného nelze očekávat u přejímaného vybavení, kde se mohou prostředky různého stupně privilegovanosti nacházet v jediném modulu.

Obecné řešení mimo rámec operačního systému asi není možné. V textu /4/ jsme popsali koncepci cílového modulu ve formě sekvenčního souboru, jehož úvodní věty definují vlastnosti všech objektů modulem zřizovaných. Jednou z vlastností exekuovatelného objektu potom může být třída viditelnosti, validovaná zaváděcím programem vůči aktuální masce viditelnosti procesu, je-li objekt volán na úrovni řídícího jazyka systému, nebo proti masce té knihovny, z níž pochází kód daný objekt volající. Jde o základní mechanismus elegantně řešící viditelnost systémových prostředků v různých prostředích. Je-li tento mechanismus k dispozici, je viditelnost procedury v rámci aplikace a její neviditelnost ve vnějším prostředí jednoduše zajistitelná v rámci jediné knihovny cílových modulů. Je ovšem třeba zamezit takovým přerušením činnosti aplikace, během nichž by mohlo být eventuálně zneužito zviditelněných prostředků mimo systém evidence implementovaný aplikačním vybavením.

Uživatelská instalace patrně nebude moci rozšířit katalog o nové typy objektů (aplikace, osoba). Implementuje-li ale katalog

takový typ objektu, kde interpretace katalogovaných dat je ponechána uživateli, lze skupinou takovýchto zobecněných objektů popsat aplikaci se všemi osobami a jejich přístupovými právy tak, aby údaje o heslech a přístupech byly dosažitelné jen z privilegovaného kódu, který realisuje i validaci hesla zcela mimo uživatelské programy. V textu /3/ jsme popsali v jiné souvislosti tento typ katalogových objektů se samodefinujícími daty ve tvaru typ-délka-údaj. Pro naše dnešní potřeby je nutno změnit samodefinující se data do podoby typ-minimální klíč privilegovanosti přistupujícího kódu-délka-údaj. Údaj bude obsahovat textovou identifikaci osoby, zakódované heslo, přístupovou masku a příznak, zda osoba již někdy do aplikace vstoupila.

K poslednímu údaji dospějeme takto: někdo zavede novou osobu a určí ji počáteční heslo. To není možno považovat za osobní dokud je jeho vlastník nezmění. Musí tak tedy učinit při prvém vstupu do aplikace. Od té chvíle nesmí být možné kódované heslo dešifrovat, takže musíme záměrně volit algoritmus neumožňující jednoznačnou zpětnou transformaci kódovaného obrazu hesla. Pro absolutní identifikaci je pak ovšem nutno obraz nově zaváděného hesla porovnat s obrazy všech hesel v danou chvíli v aplikaci známých a v případě totožnosti s některým existujícím obrazem nově zaváděné heslo zamítnout. Není na škodu upozornit na důsledek absolutní identifikace: zapomene-li osoba své heslo, musí být z aplikace vyňata a poté zavedena znovu. Stane-li se to tomu, kdo má právo osoby do aplikace zavádět, pak je-li jediný, nezbude než zrušit veškerou katalogovou informaci o aplikaci a zavést ji znovu.

Poslední tvrzení je pravdivé tehdy, nejsou-li aplikace a osoby implementovány přímo operačním systémem, který většinou ponechává privilegovaná zadní vrátka pro nápravu, samozřejmě na úkor snížené bezpečnosti celého systému. Ztráta hesla u vlastníka souborů a aplikací nemůže totiž být řešena prostou likvidací jeho instance bez možnosti rekatalogování jeho datových souborů (a patrně i jiných katalogových objektů). O exportu katalogu jsme se již dříve zmínili. Aby tyto pracné operace nemusely vůbec být prováděny, porušují implementátoři operačních systémů zásadu naprosté privatisace objektů a dávají k disposici privilegované prostředky pro změnu vlastníka

procesu a možnost ignorování přístupových práv. Je-li tomu tak, je zde další víc než pádný argument pro uživatelské doplnění operačního systému takové, že všechny privilegované činnosti budou strukturovaně zahrnuty do jedné či více aplikací s osobní identifikací (s řádnou evidencí a hmotnou odpovědností).

Katalogová data o osobě v aplikaci lze rozšířit např. o časový údaj o posledním vstupu osoby do aplikace. Protože identifikace osoby je postupována aplikačnímu vybavení, lze pohodlně doplnit strategicky významná uživatelská data o identifikaci času a osoby naposledy je aktualisující.

Reálná je i představa, že řídící struktura aplikace, kde je v každé koncové větvi stromu nabídek výkonný uživatelský program, je interaktivně popisována a generována do podoby cílového modulu, který slouží od založení aplikace jak k ladění jednotlivých větví, tak k současnému užívání větví odladěných, přičemž i neexistující koncový program je řídícím modulem přijatelně simulován.

Potřebné programové vybavení by tedy mělo obsahovat minimálně pět základních složek: privilegovanou vrstvu katalogových operací, servisní vrstvu pro řízení formátovaných obrazovkových interakcí, subsystém zavádění osob a jejich přístupových práv do aplikací, systém interaktivního popisu strukturované aplikace a generování řídícího modulu, systém pro vedení evidence v rámci aplikace. Druhou a pátou složku jsme v předchozím výkladu blíže nespecifikovali.

Implementací v prostředích bez katalogu a tříd viditelnosti exekuovatelných objektů ztrácí uvedená koncepce na celistvosti. Protože však zavádí jednotný způsob řízení a formátování interakcí a jednotný systém evidence uživatelského dění, může být i méně universální podobě aplikací s osobní identifikací jak nástrojem k zavedení uživatelské disciplinovanosti tak prostředkem unifikace programového vybavení.

Smysl příspěvku nechť čtenář nehledá v návodu "jak na to" pro zástupy potenciálních implementátorů, nýbrž v připomenutí aktuální uživatelské problematiky, jež by neměla zůstat bez odezvy u domácích výrobců programového vybavení.

Citovaná literatura :

/1/ P. Hanzálek, J. Hřebíček, Návrh struktury konverzačního programového systému pro VTV, Programování '85

/2/ Z. Rusín, Výpočetní systémy příštích let a jejich dopad na profesní sféru, Programování '82

/3/ Z. Rusín, Užití databázových přístupů v řízení dávkového zpracování úloh HZD a ve VTV aplikacích, Programování '83

/4/ Z. Rusín, Kompilační, testovací a diagnostické prostředky v interaktivním prostředí, Programování '84

/5/ Z. Rusín, Systémová podpora komunikačních úloh, Programování '85