7. Podstawowe problemy bezpieczeństwa systemów operacyjnych

Bieżący moduł przedstawia wybrane zagadnienia bezpieczeństwa dotyczące systemów operacyjnych. Przedstawione zostaną naruszenia bezpieczeństwa systemu operacyjnego, w tym w szczególności typowe formy ataku na system operacyjny oraz komponenty systemu szczególnie podatne na ataki. Omówione zostaną również problemy uwierzytelniania i kontroli dostępu, zagadnienia ochrony antywirusowej oraz zagrożenia związane z zamaskowanymi kanałami komunikacyjnymi.

Naruszenia bezpieczeństwa systemu operacyjnego przybierają różne formy. Typowo związane są z nimi następujące zagrożenia:

- włamania i kradzieże danych
- destrukcja systemu operacyjnego lub jego komponentów czy aplikacji
- wykorzystanie systemu operacyjnego do realizacji ataku na inny cel (jako *zombie*)

Różne formy ataku posiadają zwykle nieco odmienne realizacje (przebieg), jednak można wyróżnić pewien ogólny scenariusz ataku na system operacyjny. Obejmuje on zwykle następujące kroki:

1. zlokalizowanie systemu do zaatakowania
2. wtargnięcie na konto legalnego użytkownika (wykorzystując brak hasła, złamanie łatwego hasła, podsłuchanie hasła)
3. wykorzystanie błędów i luk w komponentach systemu lub w ich konfiguracji w celu uzyskania dostępu do konta uprzywilejowanego
4. wykonanie nieuprawnionych działań
5. zainstalowanie furtki dla bieżącego lub przyszłego wykorzystania
6. zatarcie śladów działalności (usunięcie zapisów z rejestrów systemowych)
7. ataki na inne komputery

Przyczyny naruszeń bezpieczeństwa tkwią najczęściej w trudności osiągnięcia pełnej kontroli nad poprawnością implementacji i konfiguracji tak złożonego i wielokomponentowego oprogramowania, jakim są współczesne systemy operacyjne. Zwykle trudności te powodują:

- błędy i luki bezpieczeństwa („dziury”) w komponentach systemu lub w ich konfiguracji
- furtki (ang. *backdoor*)
- konie trojańskie
- wirusy oraz bomby logiczne i czasowe

Jako konkretne przykłady najczęstszych obszarów błędów implementacyjnych możemy wymienić chociażby sieciowe komponenty systemu, np.:

- stos TCP/IP
  - przykładem może być atak Bonk na implementację stosu protokołów TCP/IP firmy Microsoft, który może spowodować awarię systemu zaatakowanego komputera

- lub NetBIOS:
  - WinNuke – atak umożliwiający wywołanie awarii starszych wersji systemu Windows przy użyciu usług systemu NetBIOS
- albo RPC:
  - RDS_Shell – metoda wykorzystania składnika Remote Data Services, należącego do MDAC (Microsoft Data Access Components), umożliwiająca zdalnemu napastnikowi uruchamianie poleceń z uprawnieniami systemowymi

Wrażliwe usługi systemowe i narzędziowe nie ograniczają się oczywiście do wymienionych wyżej przypadków i obejmują szerokim kręgiem m.in. usługi informacyjne (jak netstat, systat, nbtstat, finger, rusers, showmount, ident), usługi konfiguracyjne (bootparam, dhcp, zdalne zarządzanie), zdalne wywołanie procdur RPC (rpcinfo, rexec), rozproszone systemy plików (np. NFS), usługi zdalnego dostępu, mechanizm domen zaufania czy usługi komunikacyjne i pocztowe

W rożnych systemach operacyjnych, a nawet w różnych ich wersjach, można zaobserwować różną podatność komponentów na poszczególne rodzaje ataków. W wielu przypadkach istnieją gotowe i łatwo dostępne narzędzia ataków skierowane na wybrane wersje systemu lub jego konkretnych komponentów. Skuteczne przeprowadzenie ataku staje się rzeczą relatywnie prostą i nie wymagającą praktycznie żadnej wiedzy technicznej. Istotna dla tej skuteczności jest niewątpliwie możliwość zdalnej detekcji systemu operacyjnego, jego wersji i dostępnych w nim komponentów (szczególnie sług dostępnych zdalnie).

http://www.insecure.org/nmap/nmap-fingerprinting-article.html

Metody rozpoznawania systemu możemy podzielić na aktywne i pasywne. Metody aktywne realizowane są poprzez inicjowanie a następnie analizowanie połączeń (na ogół specjalnie spreparowanych) i są wobec tego skierowane wobec konkretnych stanowisk. Natomiast pasywne metody realizowane są poprzez podsłuch pakietów pochodzących z analizowanego systemu. Metody pasywne są naturalnie dużo mniej precyzyjne niż aktywne, lecz są trudno wykrywalne i mogą być skierowane przeciw celom jeszcze nie określonym ostatecznie (np. całej sieci).

Do zdalnego rozpoznawania systemu operacyjnego najczęściej wykorzystywane są usługi informacyjne dostępne w rozpoznawanym systemie oraz implementacje stosu TCP/IP.

Usługi informacyjne, jak np. ident oferują typowo charakterystyczne powitanie (*banner*), choć nie tylko informacyjne: dobrym przykładem może być serwer usługi pocztowej sendmail. Powitanie bardzo często zawiera informacje o typie i dokładnej wersji systemu operacyjnego, wersji usługi (co też istotnie przyczynia się do wyboru skutecznego narzędzia ataku). Szczęśliwie problem powitań jest już dobrze znany i coraz więcej usług stara się unikać publicznego podawania newralgicznych informacji o systemie. Z tych też powodów, często świadomie rezygnuje się z oferowania takich nadmiernie „gadatliwych" usług lub ogranicza się zdalny dostęp do nich. Osobnym sposobem ograniczania zagrożenia jest kamuflaż, polegający na spreparowaniu celowo nieprawdziwych informacji w powitaniu, co nie jest niestety gwarancją ochrony systemu przed wprawnym intruzem, jednak może utrudnić i wydłużyć realizację ataku (*security trough obscurity*).

W przypadku stosu TCP/IP najskuteczniejszą aktualnie metodą zdiagnozowania typu i wersji systemu okazuje się być analizowanie zachowania protokołu implementacji TCP w przypadku retransmisji – tzw. badanie RTO. Badanie RTO (*Retransmission Time-Out*) jest to pomiar czasów pomiędzy retransmisjami pakietów SYN+ACK w trakcie nawiązywania odpowiednio spreparowanego przez atakującego połączenia. Badanie obejmuje ilość retransmisji i odstępy między nimi, co, jak się okazuje, należy do „cech osobniczych" systemu operacyjnego.

Przykładowo system MacOS X wysyła po 5 nieskutecznych retransmisjach segment RST, a systemy Windows i Linux milcząco zamykają wpółotwarte połączenie, różniąc się istotnie liczbą i interwałami retransmisji.

Również w przypadku tej metody detekcji systemu stosować można kamuflaż. Przykladem zaawansowanego narzędzia do kamuflażu jest łata *stealth patch* na jądro systemu Linux, która m.in.:

- blokuje nieprawidłowe pakiety ACK
- blokuje pakiety z flagami SYN i FIN
- blokuje pakiety z niepoprawnymi flagami i kombinacjami flag
- blokuje pakiety ICMP (za wyjątkiem echo)

Podobną funkcjonalność ma pakiet IP Personality – zestaw łat na jądro Linuksa i iptables, niestety już nierozwijany.

Uwierzytelnianie

Jednym z najistotniejszych mechanizmów ochrony systemu operacyjnego jest uwierzytelnianie użytkowników, niezbędne dla określania ich uprawnień oraz zdiagnozowania ewentualnej próby niepowołanego dostępu.

Uwierzytelnianie w systemie Unix/Linux

W większości systemów operacyjnych, również w systemach z rodziny Unix/Linux, podstawowym narzędziem uwierzytelniania jest weryfikacja hasła użytkownika. W tym punkcie przedstawimy najpierw klasyczny mechanizm przechowywania haseł w systemie Unix.

Działanie klasycznego mechanizmu tworzenia i rejestrowania w systemie Unix hasła przebiega następująco:

- użytkownik wybiera 8 znakowe hasło
- hasło jest zamieniane na 56b ciąg za pomocą 7b kodu ASCII
- powstały 56b ciąg jest kluczem algorytmu DES
- e-blok algorytmu DES jest modyfikowany za pomocą 12b wartości – ziarna (ang. *salt*) ustalanego na ogół na podstawie bieżącego czasu
- tak zmodyfikowany algorytm DES jest wykonywany na 64b bloku złożonym z samych zer
- wyjście podaje się na wejście kolejnej iteracji (25 iteracji)
- 64b wynik transformowany na 11-znaków z alfabetu 64 znakowego (A-Z, a-z, 0-9, '.', '/')

Wynikowa postać hasła jest zapamiętywania w pliku konfiguracyjnym (klasycznie był to /etc/passwd) i każdorazowo porównywana przez narzędzie login, rejestrujące sesję użytkownika w systemie, z przetransformowanym hasłem wprowadzanym przez logującego się użytkownika.

Jak wiemy, mechanizm haseł jest podatny na problem złamania hasła. W przypadku klasycznego mechanizmu uwierzytelniania metoda przeszukiwania wyczerpującego (*brut-force attack*) może być przykładowo wykonana następująco:

- 8 znaków z 36-znakowego alfabetu daje 36^8 czyli ok. 2,8 biliona kombinacji
- załóżmy moc obliczeniową o wydajności 6,4 miliona iteracji DES na sekundę
- 25 iteracji dla każdej kombinacji hasła – 1 milion kombinacji w 4 sek.
- dowolne hasło zostanie złamane (2,8 biliona kombinacji) w 4 miesiące

Przykład ten, aczkolwiek hipotetyczny, jednak obnaża wyraźnie tempo starzenia się haseł oraz uzmysławia dobitnie konieczność i doniosłość systematycznych zmian hasła dla wszystkich kont w systemie operacyjnym.

Wobec problemu złamania hasła, we współczesnych systemach z rodziny Unix/Linux stosuje się pewne usprawnienia. Obejmują one dodatkowo ochronę haseł przed ich pozyskaniem (w celu utrudnienia przeszukiwania wyczerpującego oraz wymuszanie odpowiednio wysokiego stopnia skomplikowania hasła (w celu utrudnienia ataku słownikowego). Ochrona haseł przed ich pozyskaniem sprowadza się do ukrycia ich postaci składowanej w systemie haseł poza dostępem zwykłego użytkownika – w przypadku systemu Unix/Linux jest to przeniesienie haseł do oddzielnej lokalizacji (pliku /etc/shadow). Realizuje to albo sam system operacyjny (konkretna dystrybucja), albo oddzielne pakiety, np. shadow-in-a-box, Linux Shadow Password Suite, itp. W praktyce okazuje się, iż czasami nadal możliwe ataki na pliki shadow (wykorzystują głównie luki w usługach, obrazy *core dump*), co umożliwia przechwycenie postaci składowanej haseł i ich późniejsze dekodowanie. Rozwiązaniem skuteczniejszym i bardziej uniwersalnym dla ochrony haseł mogą być zatem centralne bazy katalogowe, np. NIS, NIS+, czy bazy dostępne poprzez protokół LDAP.

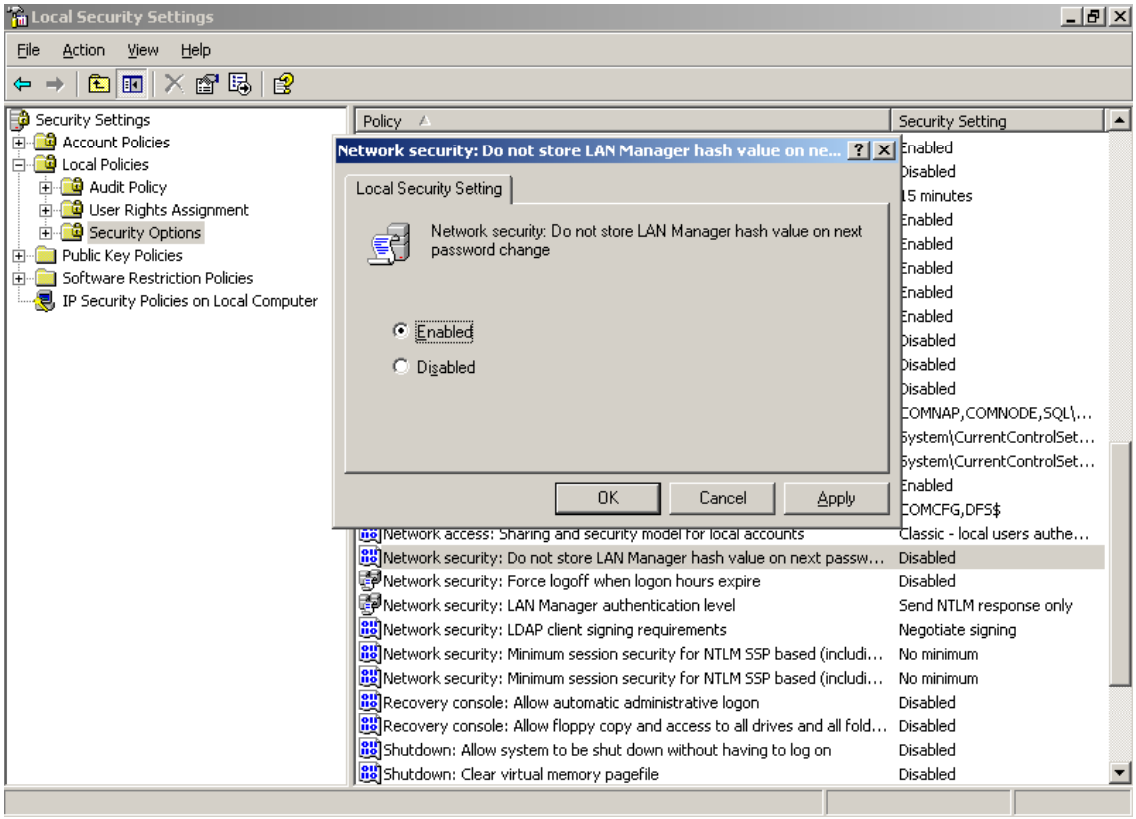
W celu kontroli poziomu trudności haseł stosuje się różnorodne testery jakości haseł uaktywniane w momencie ustawiania nowego hasła przez użytkownika. Często spotykane są np. passwd+ (zastępuje passwd) lub anlpasswd, czy npasswd.

W bieżących wersjach systemów tej rodziny odchodzi się od wykorzystania algorytmu DES na korzyść silniejszych mechanizmów kryptograficznych i pozwalających wybierać hasła bez górnego limitu znaków (lub z wysokim limitem), np. z wykorzystaniem algorytmów MD5 czy Blowfish.

Uwierzytelnianie w systemie MS Windows NT 4.0

W systemie MS Windows NT 4.0 użytkownik otrzymuje losowo wygenerowany SID (Security ID). Procedura logowania wymaga wywołania przerwania sprzętowego (poprzez kombinację Ctrl-Alt-Del), co ułatwia kontrolę nad poprawnym wywołaniem właściwego programu logującego. Ze względu na złożoną ewolucję systemu Windows, hasła w systemie są przechowywane w różnych postaciach. Niezakodowane hasła, przetwarzane przez system, przechowywane są w zastrzeżonym obszarze LSA (Local Security Authority) rejestru: `HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets`, dostępnym tylko dla usługi Security Accounts Manager. Od wersji NT 4.0 hasła mogą być również zakodowane (w postaci tzw. *hash*) funkcją MD5 z ziarnem i z 40b kluczem RSA i przechowywane w rejestrze: `HKLM\SAM` oraz w zastrzeżonym obszarze systemu plików NTFS: `%SYSTEMROOT%\SYSTEM32\CONFIG\SAM`. Jednak dla zachowania zgodności wstecz (z linią 9x) obok postaci hash MD5 umieszczane są kopie haseł – LM hash – zakodowane autorskim algorytmem LanMan (z protokołu NT Lan Manager), bez ziarna. Postać ta jest dużo mniej bezpieczna i jej dostępność drastycznie podnosi skuteczność złamania hasła. Na szczęście w

systemie Windows przechowywanie postaci LM hash można wyłączyć. Rysunek 1 przedstawia obraz ekranu narzędzia Zasad lokalnych zabezpieczeń, które umożliwia uaktywnienie wyłączenia LM hash – tak, widoczną na ekranie opcję trzeba włączyć, aby postać LM hash nie była tworzona i to dopiero od chwili następnej zmiany hasła (do kiedy to dotychczasowe LM hashe pozostają w systemie).



Rysunek 1. Uaktywnienie opcji wyłączenia przechowywania postaci LM hash haseł

W poprawkach serwisowych SP2 i SP3 do systemu wprowadzono następujące usprawnienia:

- opcję *syskey* (SP2) – postacie hash można zaszyfrować kluczem SYSTEM KEY (128b klucz RSA zamiast klucza 40b) co znacznie utrudnia łamanie haseł (jednak nadal dostępne są gotowe narzędzia wyłuskiwania i łamania haseł, choć bardziej złożone niż uprzednio)
- filtr słabych haseł Passfilt.dll (SP3)
- dodatkowe restrykcje
  - ograniczenia czasowe (pory dnia, data ważności konta)
  - ograniczenia stanowisk logowania
  - limit ilości błędnych uwierzytelnień lub korzystania z nieaktualnego hasła
- rozszerzenia procedury uwierzytelniania

- API opublikowane przez Microsoft umożliwia wykorzystanie inteligentnych kart uwierzytelniających lub biometrii

Niestety ciągle istnieje w module zarządzania hasłami wiele luk i dostępnych jest wiele gotowych narzędzi *exploit* (wykorzystują uprawnienia administracyjne do importu plików SAM, luki w systemie tworzenia kopii zapasowych systemu RDISK, itp.)

Uwierzytelnianie w systemie MS Windows 2000, XP, 2003

W systemie MS Windows 2000 i jego następcach algorytm uwierzytelniania NTLM został zastąpiony przez Kerberos TGP. Ponadto opcja syskey jest domyślnie włączona, a przechowywanie postaci LM hash jest domyślnie wyłączone. Należy jednak mieć na uwadze, iż w środowiskach heterogenicznych istniejące starsze wersje Windows nie obsługują Kerberosa, stąd mogą nadal wymagać obsługi LM hash, co osłabia bezpieczeństwo całego srodowiska.

Uwierzytelnianie w systemie Novell NetWare

W systemie Novell NetWare stosuje się od wczesnych jego wersji dość wyrafinowaną I powszechnie uważaną za bezpieczną procedurę uwierzytelniania. Wykorzystuje ona mechanizmy kryptograficzne i przebiega w uproszczeniu następująco:

- w bazie katalogowej NDS (NetWare Directory Service) przechowywany jest skrót hasła użytkownika z ziarnem (jest nim identyfikator użytkownika) oraz para kluczy RSA
- najpierw stacja sieciowa uwierzytelnia się w imieniu użytkownika wobec wybranego serwera (NetWare od wersji 4 stosuje SSO) metodą zawołanie-odzew
- następnie serwer przesyła klucz publiczny NDS (KNDS), a stacja losuje klucz jednorazowy K1, który prześle serwerowi zaszyfrowany kluczem KNDS
- K1 posłuży serwerowi do bezpiecznego przekazania klucza prywatnego RSA użytkownika
- stacja przekształca klucz RSA do postaci GQ – Gillou-Quisquater (asymetryczny algorytm uwierzytelniania – NetWare od wersji 4 nie stosuje RSA) i natychmiast usuwa z pamięci hasło wraz z kluczem RSA
- klucz GQ posłuży do uwierzytelniania użytkownika w dostępie do zasobów sieci

Relacja zaufania

Jak wiemy z poprzednich modułów, pewnym ograniczeniem ryzyka pozyskania hasła przez intruza, jest zastosowanie mechanizmu SSO (*Single Sign-On*). Systemy Unix/Linux i Windows (w konfiguracji domenowej, od wersji NT Advanced Server) pozwalają wykorzystać mechanizm SSO w postaci tzw. **relacji zaufania**. Dzięki jej istnieniu uwierzytelniony użytkownik systemu (domeny) **zaufanego** może mieć dostęp do zasobów systemu **ufającego** bez konieczności ponownego uwierzytelniania. A wobec tego, hasło nie zostaje ponownie przesyłane ani przetwarzane przy zdalnym dostępie do kolejnego systemu operacyjnego. Relacje zaufania mogą mieć charakter jednostronny lub dwustronny i, co ważne, nie są przechodnie.

Prawa dostępu do zasobów

Standard POSIX (*Portable Operating System Interface*) 1003.1

Standard POSIX 1003.1 jest powszechnie wspierany przez współczesne systemy operacyjne. Wymaga on obsługi następujących elementów procesu autoryzacji i kontroli dostępu do zasobów:

- prawa: r (read – odczyt), w (write – zapis), x (execute – wykonanie)
- kategorie użytkowników: u (user – właściciel zasobu), g (group), o (others)
- dodatkowo prawa Set User Id, Set Group Id, Sticky, znane z systemu Unix.

Niektóre aplikacje oferują własne rozszerzenia tego modelu uprawnień, np. ProFTPD (http://proftpd.linux.co.uk/). Również w samych systemach operacyjnych spotyka się rozszerzone implementacje modelu POSIX. Rozszerzenia te obejmują na ogół:

- listy kontroli dostępu ACL: np. w systemach AIX, Solaris
- model ścisłej kontroli dostępu MAC: w systemach Trusted Solaris, Trusted IRIX, Ultrix, HP-UX czy Xenix.

Standard POSIX 1003.1e/1003.2c

Ponieważ wcześniejsza wersja tego standardu dobrze odpowiadała w praktyce tylko klasycznym systemom Unix i w wielu nowszych systemach implementowano różnorodne rozszerzenia standardu, zaproponowano nowe wersje standardu. obejmujące m.in.:

- mechanizmy ACL, CAP, RBAC, MAC
- audyt kontroli dostępu

W implementacji nowego standardu prym wiedzie projekt TrustedBSD (http://www.trustedbsd.org/) – dla systemu FreeBSD. Dostępne są również implementacje dużej części standardu w systemie Linux z jądrem w wersjach od 2.5.46 (do jądra 2.4 jest łata obsługująca ACL standardowo dystrybuowana np. w SuSE Linux). Standard POSIX 1003.1e/1003.2c wspierają różne systemy plików Ext2, Ext3, IBM JFS, ReiserFS oraz SGI XFS. Dostępne jest też wsparcie NFS – w wersji NFSv3 możliwe jest przekazywanie list ACL (chociaż brak specyfikacji metody przekazywania skutkuje różnymi implementacjami), a wersja NFSv4 posiada mechanizm NFS ACL, lecz niestety niezgodny z POSIX. Istotne jest zachowanie wymagań standardu również w przypadku takich operacji na zasobach jak archiwizacja i tworzenie kopii zapasowych – tu popularne jest chociażby narzędzie *pax* (*portable archive interchange*), które zachowuje wszystkie wymagane standardem parametry.

Listy dostępu ACL

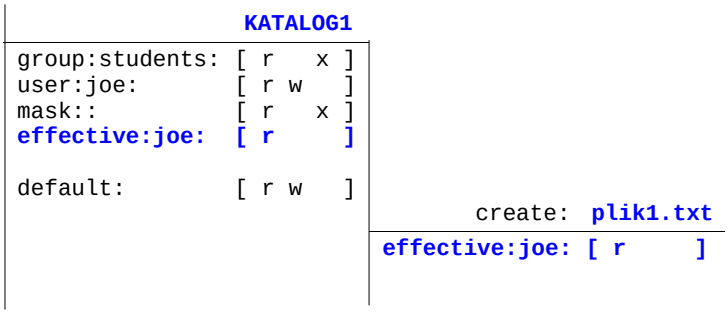
Listy ACL są w praktyce dostępne we wszystkich popularnych współcześnie systemach.

ACL w jądrze Linux

System Linux obsługuje dwa rodzaje uprawnień ACL:

- *minimal* ACL – prawa r w x dla u g o
- *extended* ACL – rozszerzone prawa i maski praw

Rysunek 2 przedstawia kombinację uprawnień do zasobu KATALOG1 dla użytkownika joe należącego do grupy students. Prawa efektywne są wyznaczane za pomącą algorytmu opisanego niżej. Uprawnienia defaults określają maskę dziedziczenia uprawnień w głąb – ustawianą tylko dla katalogów i dotyczącą tylko nowotworzonych obiektów.



Rysunek 2. Schemat wyznaczania uprawnień mechanizmu ACL w systemie Linux

Rysunek 3 przedstawia przykład wykorzystania narzędzi systemu Linux operujących na uprawnieniach extended ACL. Są to polecenia *getfacl* i *setfacl.*

```
$ setfacl -d -m group:students:r-x katalog1
$ getfacl --omit-header katalog1
  user::rwx
  user:joe:r-x
  group::r-x
  mask::r-x
  other::---
  default:user::rwx
  default:group::r-x
  default:group:students:r-x
  default:mask::r-x
  default:other::---
```

Rysunek 3. Przykładowe wywołanie poleceń *getfacl* i *setfacl* w systemie Linux

System operacyjny weryfikuje po kolei kategorie uprawnionych podmiotów (właściciel zasobu, grupa, wyróżnieni użytkownicy), aby określić, do której kategorii należy podmiot żądający zasobu i jakie prawa z listy ACL należy zaaplikować. Kolejność weryfikacji kategorii uprawnień w celu określenia praw z listy ACL jest następująca:

1. właściciel zasobu (***u**ser*)
2. wyszczególniony użytkownik (filtrowane przez maskę)
3. grupa zdefiniowana (jeśli zawiera żądane prawa filtrowane przez maskę)
4. dowolna z wyszczególnionych grup (jeśli zawiera żądane prawa filtrowane przez maskę)
5. jeśli żadna dopasowana grupa nie zawiera praw – żądanie jest odrzucane
6. pozostali użytkownicy (***o**thers*) bez maski

Implementacje ACL w systemie Unix

W systemach z rodziny Unix, listy ACL mogą być przechowywane na różne sposoby. Zwykle definicje ACL przechowuje:

- węzeł przysłonięty (*shadow inode*) – wiele i-węzłów może być związanych z tym samym węzłem przysłoniętym (Solaris UFS file system)
- rozszerzone atrybuty EA obiektów (*Extended Attributes*) – są to struktury informacyjne związane z plikami i in. obiektami w systemie (Linux)

Maksymalna ilość pozycji możliwych do przechowania na liście ACL zależy oczywiście od miejsca przechowywania. Ilość tę dla wybranych systemów plików przedstawia rysunek 4.

system plików	max. ilość pozycji
XFS	25
Ext2, Ext3	32
ReiserFS, JFS	8191

Rysunek 4. Przykładowe wywołanie poleceń *getfacl* i *setfacl* w systemie Linux

Prawa dostępu ACL w MS Windows

W systemie Windows wprowadzono mechanizm ACL (odbiegający od POSIX) w systemie plików NTFS. Lista możliwych praw i kategorii uprawnionych podmiotów jest imponująco duża. Wśród ciekawych praw można znaleźć takie atrybuty jak np. append, delete, change permissions, take ownership, change ownership, i in. (istnieją podobne atrybuty dla niektórych systemów plików z rodziny Unix, np. w systemie Linux narzędzie *chattr* oferuje takie atrybuty dla systemu plików Ext2/Ext3). Dzięki oddzielnie stosowanym przypisaniom GRANT i unieważnieniom DENY praw możliwa jest do osiągnięcia duża elastyczność konfiguracji uprawnień. W definicji uprawnień na liście można wykorzystywać statyczne oraz dynamiczne grupy użytkowników (np. *All Authenticated Users*). Ponadto Windows oferuje dla ACL wsparcie protokołu CIFS (*Common Internet File System*) umożliwiając tym samym tworzenie rozproszonych systemów plików uwzględniających definicje ACL. Z kolei niezależny pakiet Samba dostępny dla systemu Linux, będący implementacją wcześniejszego protokołu współdzielenia zasobów w Windows – SMB, oferuje listy ACL zgodne z POSIX.

Uprawnienia specjalne w systemie Unix

Flaga *suid*

W systemach z rodziny Unix wykorzystuwana jest często flaga *suid* = *Set User Id* oznaczająca specjalne uprawnienie do przejmowania przez proces (uruchomiony z programu z takim atrybutem) praw dostępu zdefiniowanych dla właściciela tego programu. Mechanizm ten jest w niektórych sytuacjach niezbędny do poprawnej pracy systemu, korzystają z niego takie programy narzędziowe jak np. *passwd*.

Niestety, mechanizm ten może stanowić potencjalne zagrożenie dla bezpieczeństwa systemu operacyjnego, zwłaszcza jeśli właścicielem programu z flagą suid jest administrator. Często spotykany atak polega na wyszukiwaniu programów z atrybutem *stuid* należących do superużytkownika *root*, w celu wykorzystania błędów lub nieodpowiedniej konfiguracji systemu i uzyskania uprawnień administracyjnych. Zatem programy z ustawionym bitem *suid* powinny być szczególnie chronione.

Podobny mechanizm, o podobnym zagrożeniu, wprowadzono również dla grupy – flaga *sgid* = *Set Group Id*.

Redukcja przywilejów (*privilege reduction*)

Procesy uruchomione z podwyższonymi uprawnieniami (głównie administracyjnymi) stanowią łakomy kąsek dla intruza. Jest to szczególnie istotne w systemie Unix, gdzie przewidziano jedno

konto administracyjne – superużytkownika *root*, które umożliwia wykonanie praktycznie wszelkich działań w systemie.

Jednak w większości przypadków, nawet ważne dla systemu procesy wymagają wysokich uprawnień tylko przez krótki okres czasu (zwykle początkowy). Koncepcja redukcji przywilejów sprowadza się do ograniczania zagrożenia przejęcia uprawnień procesu mogącego stać się potencjalnym celem ataku. Ograniczenie zagrożenia osiągane jest poprzez świadomą rezygnację z wyższych uprawnień (administracyjnych) natychmiast gdy newralgiczne operacje zostaną przez proces zakończone. W czasie dalszej pracy proces działa z niskimi (możliwie najniższymi) uprawnieniami.

Innym sposobem ograniczenia zagrożenia, jest wyodrębnienie wielu szczegółowych uprawnień administracyjnych i przydzielanie do procesu tylko niezbędnych, zamiast wszechwładnego prawa superużytkownika. Standard POSIX przewiduje taki mechanizm nazywany *capabilities* (CAP), dostępny w systemie Linux od jądra 2.2 (ftp://linux.kernel.org/pub/linux/libs/security/linux-privs). CAP pozwala na rozdzielenie uprawnień administracyjnych superużytkownika *root* na zbiór szczegółowych uprawnień i powiązanie ich z systemem plików poprzez dodatkowe bity praw dostępu.

Popularną implementacją mechanizmu CAP w systemie Linux jest narzędzie LCAP (http://pweb.netcom.com/~spoon/lcap/). Oferuje on m.in. następujące uprawnienia szczegółowe:

- administrowanie modułami jądra
- administrowanie siecią
- dowiązywanie do gniazd numerów portów zarezerwowanych dla serwisów systemowych
- realizacja komunikacji rozgłoszeniowej i grupowej w sieci
- omijanie ograniczeń dotyczących kontroli dostępu do plików
- zmiana informacji o właścicielu i grupie plików
- kontrola plików specjalnego rodzaju
- kontrola flag *suid* oraz *sgid*
- omijanie ograniczeń dotyczących wysyłania sygnałów do procesów
- blokowanie stron w pamięci fizycznej
- omijanie limitów zasobowych

Separacja przywilejów (*privilege separation*)

Kolejnym sposobem ograniczania zagrożeń związanych z nadużyciem praw dostępu jest separacja przywilejów posiadanych przez uruchomiony proces. Koncepcja separacji przywilejów polega na wyodrębnieniu zadań wymagających wysokich uprawnień w postaci odrębnego procesu. Błędy w programie procesu głównego, działającego z uprawnieniami zwykłego użytkownika, nie umożliwiają już tak łatwego uzyskania wysokich uprawnień. Istotne jest, iż ta koncepcja nadaje również się do systemów z klasycznym przydziałem uprawnień administracyjnych (bez CAP). Przykład sztandarowy wykorzystania separacji przywilejów to m.in. OpenSSH.

Malware

Pod pojęciem *malware* należy rozumieć wszelkie niepożądane (złośliwe) w systemie oprogramowanie, którego pojawienie się w systemie było niezamierzone i działanie jego przynosi systemowi wymierne straty. Infekcje systemu operacyjnego mogą być spowodowane różnymi typami oprogramowania malware, do którego należą:

- wirusy
- konie trojańskie
- spyware
- dialery

Poniżej zajmiemy się przedstawieniem pierwszej klasy oprogramowania malware – wirusami.

Wirusy i inne robactwo

Wirus (łac. *virus* – trucizna) to – najprościej definiując – kod samopowielający się w systemie operacyjnym. Istnieje wiele bardzo odmiennych rodzajów wirusów w tym w szczególności tak specyficzne jak:

- wirusy skryptowe i w makrodefinicjach (np. w dokumentach edytorów tekstu)
- wirusy sieciowe (*worms*) – przenoszone poprzez sieć (uruchamiane lokalnie)
- wirusy w poczcie elektronicznej (przesyłane jako załączniki)
- wirusy atakujące programy pocztowe (np. Win.Redteam), które przenoszą się tradycyjnie (przez system operacyjny), a jako ofiary wybierają klientów poczty elektronicznej (np. Eudorę) – zarażają klienta, wykonując makrodefinicje i rozsyłając się do wszystkich osób z listy adresowej.
- wirusy bujdy (hoax-viruses) – udające ostrzeżenia zwykłe żarty rozpowszechniane pocztą elektroniczną (często przez nieświadomych użytkowników).

Efekty działania wirusów to najczęściej:

- utrudnienie pracy systemu (zużywanie zasobów: CPU, pamięci, przestrzeni dyskowej, pasma sieci)
- destrukcja danych
- wyciek danych na zewnątrz (np. poprzez zamaskowany kanał komunikacyjny).

Obserwacja ewolucji oprogramowania wirusowego w ostatnich latach skłania do wyróżnienia pewnych trendów. Można przykładowo wybrać następujące obserwacje:

- 2002 r.: pojawiają się pierwsze wirusy wieloplatformowe, atakujące pliki wykonywalne w popularnych formatach zarówno exe (MS Windows) oraz elf (Linux)
- 2003 r.: wirusy typu *blended threats* – łączą cechy różnych klas: wirusów, koni trojańskich, robaków sieciowych (np. MS Blaster)
- 2004 r.: ataki na systemy wbudowane: np. wirus nazwany Ratter atakujący Windows CE, wirus Cabir atakujący telefony komórkowe z systemem Symbian

- 2005 r.: wzmaga się aktywność wirusowa w systemach wbudowanych – pojawił się Lasco, wirus telefonów komórkowych z możliwością infekowania plików (od 2005 McAfee i TrendMicro oferują antywirusy dla systemów mobilnych)

W oprogramowaniu wirusowym pojawiają się zaawansowane mechanizmy, takie jak kamuflaż i techniki anty-antywirusowe:

- polimorfizm
- metamorfizm
- maskowanie (*stealth*) oraz opancerzenie (*armor*) – techniki różnorodnej ochrony kodu, np. uniemożliwienie debugowania.

Wzmaga się aktywność retrowirusów stosujących obronę agresywną przed detekcją, poprzez ataki skierowane przeciw oprogramowaniu antywirusowemu. Najbardziej zaawansowane wirusy sieciowe posiadają strukturę modułową, w której występują moduły zdalnie aktualizowane (przez Internet).

Oprogramowanie malware może stosować też przeróżne sztuczki w celu ukrycia się w systemie plików i uniemożliwienia skutecznego wyszukania przez oprogramowanie antywirusowe. W tym kontekście ciekawe możliwości oferuje, dotąd mało znany i wykorzystywany mechanizm ADS (*Alternate Data Stream*) w systemie NTFS. Teoretycznie miał on zapewniać wsparcie międzyplatformowe (np. z systemem Macintosh). Jednak istotne z naszego punktu widzenia jest to, iż system Windows w bieżących wersjach pozwala tworzyć pliki (programy) w ADS, choć sam nie zwraca na nie uwagi (nawet ich nie pokazuje), co z kolei pozwala skutecznie ukrywać malware przed użytkownikiem, a nawet przed większością skanerów antywirusowych.

Nakłady poniesione na usunięcie infekcji i zwalczanie skutków są całkiem pokaźne. Przykładowo wg Computer Economics (2001r.) w roku 1999 osiągnęły w sumie 12,1 mld USD, a w roku 2000 – w sumie 17,1 mld USD. Jedne z najkosztowniejszych pod tym względem wirusów w historii to z pewnością:

- Love Bug (łącznie ok. 50 odmian), 2000 r.: 8,7 mld USD
- Code Red, 2001 r.: 2,6 mld USD
- Melissa, 1999 r.: 1,2 mld USD
- Explorer, 1999 r.: 1 mld USD
- SirCom, 2001 r.: 460 mln USD

Ochrona przed wirusami

Ochrona przed wirusami jest realizowana przez różnorodne oprogramowanie nazywane ogólnie antywirusowym, a obejmujące jedną lub kilka z wymienionych poniżej kategorii.

1. Programy wyszukujące programy zarażone (skanery)

   przed ich uruchomieniem

   - wymagające znajomości kodu wirusa
     - wyszukujące identyfikator wirusa (*fingerprint*)
     - wyszukujące fragment charakterystyczny wirusa

- nie wymagające znajomości kodu wirusa
  - stosujące analizę zmian struktury programu

na podstawie efektów

- metodą "przynęty"
- poprzez analizę zmian w systemie plików po zakończeniu programu

2. Programy wzbogacające możliwości ochronne systemu
   - monitory operacji wykonywanych w systemie wyszukujące działania wirusa albo w trakcie instalacji lub powielenia wirusa, albo w trakcie wykonywania zadań
     - destruktywnych
     - demonstracyjnych
   - skanery usług sieciowych (np. poczty elektronicznej, WWW)
   - system plików zabezpieczony przed zapisem
3. Programy usuwające:
   - kod wirusa
   - skutki działania wirusa

Poniżej wymienione są przykładowe narzędzia ochrony przed malware.

Zdalne skanery internetowe

- http://skaner.mks.com.pl
- http://hausesall.trendmicro.com
- http://www.bitdefender.com
- http://www.pandasoftware.com/activescan/pol/activatescan_principal.htm
- http://www.kaspersky.pl/serwices.html/s=online_vir_chk

Programy anty-spyware

- Ad-aware: http://www.snapfiles.com/get/adaware.html
- Spybot: http://www.safer-networking.org/pl/download/
- Bazooka: http://www.kephyr.com
- SpywareBlaster: http://javacoolsoftware.com/downloads.html

Zamaskowane kanały komunikacji

Zamaskowane kanały komunikacji (*covered channels*) są pojęciem ściśle powiązanym z malware, choć nie ograniczają się wyłącznie do tego przypadku. Poprzez zamaskowany kanał komunikacji (*covered channel*) rozumie się wykorzystanie w celu przekazania informacji takiego mechanizmu systemu operacyjnego, które został opracowany w celu nie związanym z

komunikacją i na ogół nie jest z nią w ogóle kojarzony. Kanał ten może być wykorzystywany do nieujawnionego przekazywania wiadomości, tak aby ukryć nie tylko treść tej wiadomości, ale i sam fakt dokonywania transmisji. Przykładami zamaskowanych kanałów komunikacji są:

- kanał czasowy (obciążenie systemu) – transmisja bitu następuje poprzez wzmożenie obciążenia systemu w danym kwancie czasu (bit 1) lub obniżenie obciążenia (bit 0)
- kolejka wydruku (wstawianie do niej i usuwanie określonego zadania wydruku)
- kanał poprzez tworzenie / usuwanie ogólnodostępnego pliku w systemie plików