Bezpieczeństwo OS Linux

System operacyjny Linux jest jednym z najbardziej popularnych oraz stosowanych w sieci systemów operacyjnych. Bardzo często jest wykorzystany jako OS różnych serwerów, narzędzi systemów wbudowanych, routerów, stacji roboczych i t.d. Linux ma dużą ilość zalet, wśród których najważniejszą i najbardziej mającą wpływ na popularność (zdaniem autora) jest ten fakt, że Linux jest darmowy.

Ochrona Systemu Operacyjnego jest jednym z najważniejszych elementów cyberbezpieczeństwa. Przyjęcie kontroli cyberprzestępcą nad OS likwiduje całkowicie możliwość bezpiecznego funkcjonowania systemu informatycznego. Nie ważne na ile odporne hasło jest stosowane dla szyfrowania danych, zainfekowany OS przechwyci hasło w chwili naboru hasła na klawiaturze. Nie jest ważna długość kluczy SSH, OS ma do niego dostęp, jak i do certyfikatów SSL.

Istnieją dwa najbardziej powszechne zdania na temat bezpieczeństwa OS Linux. Niektórzy uważają, że nie jest to system bezpieczny, ponieważ jest produktem rozproszonego zdecentralizowanego zespołu programistów z całego świata, a otwarty kod źródłowy pozwala przestępcom znaleźć błędy w kodzie i opracować exploity dla nich. Inni uważają, że OS Linux jest bardzo bezpiecznym systemem i w ogóle nie jest podatny na ataki hakerów, nawet nie wymaga on instalacji programu antywirusowego. Względem autora, obaj zdania są nieprawidłowe. Mimo tego, że OS Linux jest wynikiem pracy rozproszonego zdecentralizowanego zespołu specjalistów, Linux jest bardziej stabilnym, niezawodnym i bezpiecznym od produktu centralizowanego zespołu programistów Microsoft. Z drugiej strony, domyślna konfiguracja OS Linux nie jest bezpieczna i odporna na ataki, jednak dysponuje narzędziami które pozwalają wielokrotne podnieść bezpieczeństwo OS.

Efektywny systemy bezpieczeństwa OS bazuje się na następnych elementach:

- Kontrola dostępu fizycznego do narzędzia z zainstalowanym OS.
- Bezpieczeństwo sieci.
- Łatanie łuk bezpieczeństwa.
- Kontrola dostępu do zasobów ze strony użytkowników systemu.
- Wykrycie niesankcjonowanego dostępu.

1. Kontrola dostępu fizycznego do narzędzia z zainstalowanym OS.

Istnieją różne metody kontroli fizycznego dostępu do narzędzia:

- secureBoot;
- instalacja hasła na UEFI;
- szyfrowanie partycyj twardego dysku;
- fizyczna izolacja narzędzia;
- organizacja ochronnych przedsięwzięć;
- inne.

2. Bezpieczeństwo sieci.

Elementami bezpieczeństwa sieciowego OS są:

- Zapora sieciowa;
- Monitorowanie sieci;
- Systemy wykrywania i zapobiegania włamaniom (intrusion detection system (IDS), intrusion prevention system (IPS)). Przykładowe systemy są: BRO i SNORT.

3. Łatanie łuk bezpieczeństwa.

Wykrycie nowych błędów w kodzie aplikacyj, systemów operacyjnych oraz protokołach komunikacji jest zdarzeniem częstym i rzeczywistym. Ważnym jest aktualizacja łatek bezpieczeństwa we właściwym czasie. Monitorowanie informacji o obecnie wykrytych podatnościach oraz exploitów dla podjęcia decyzji o hamowaniu podatnych na ataki usług i aplikacyj.

4. Kontrola dostępu do zasobów ze strony użytkowników systemu.

Standardowym mechanizmem kontroli dostępu w OS Linux jest model Uznaniowej Kontroli Dostępu (discretionary access control (DAC)). Ten model jest realizowany w sposób ustawienia dla każdego pliku odpowiedniego kodu trybu dostępu do pliku. Każdy plik ma:

- flaga rodzaju pliku,
- tryb dostępu dla właściciela,
- tryb dostępu dla grupy właścicieli,
- tryb dostępu dla pozostałych użytkowników,
- właściciela pliku,
- grupę właściciela pliku.

Flaga rodzaju pliku może mieć następne znaczenia:

- „-” - Brak flagi.
- „l” - Dowiązanie symboliczne (symbolic link)
- „d” - Katalog (directory)
- „b” - Urządzenie blokowe (block device)
- „c” - Urządzenie znakowe (character device)
- „p” - Kanał, urządzenie fifo (fifo device)
- „s” - Unix sokiet (unix domain socket)

Tryby dostępu dla właściciela, grupy właścicieli i pozostałych użytkowników mogą być następne:

- „r” - zezwolenie na odczyt (**r**ead),
- „w” - zezwolenie na wpis, na edycję pliku (**w**rite),
- „x” - zezwolenie na uruchomienie (e**x**ecute),
- kombinacja tych znaczeń. Przykładowe „r-x”: można czytać i uruchomić, ale nie można edytować.

Właściciel pliku:
Ma odpowiadać jednemu z obecnych w systemie użytkowników.
Informację o użytkownikach w systemie Linux Debian można wyświetlić poleceniem:

```
test@test-lab:~$ cat /etc/passwd
```

Grupa właścicieli pliku:
Ma odpowiadać jednej z obecnych w systemie grup użytkowników.
Informację o grupach użytkowników w systemie Linux Debian można wyświetlić poleceniem:

```
test@test-lab:~$ cat /etc/group
```

Przykład uprawnień dostępu, (polecenie *ls -lh*)

```
test@test-lab:~$ ls -lh
total 461M
-rw-r--r--    1 test   users  49M   Sep    29 11:23  750039.pdf
---x------    1 root   root   64    Mar    12  2018  cachedrop.sh
drwxrwxr-x    5 test   users  412M  Mar    12  2018  DevelopAppAndroid
```

Pierwszy symbol w wierszu jest flagą (dla *DevelopAppAndroid* to jest „*d*” - Katalog, dla pozostałych plików „-” - flagi nie ma).

Następne trzy symbole określają uprawienia właściciela pliku (Dla *DevelopAppAndroid* „*rwx*” - odczyt, wpis, uruchomienie; dla *750039.pdf* to jest „*rw-*” - odczyt, wpis; dla *cachedrop.sh* to jest „*--x*” - można tylko uruchomić.

Następne trzy symbole określają uprawienia grupy właścicieli.

Ostatnie trzy symbole określają uprawienia wszystkich pozostałych użytkowników.

Plik „*750039.pdf*”:

- flag: „-” - nie ma flagi.
- tryb dostępu dla właściciela: „*rw-*” właściciel może czytać plik i robić wpis do pliku, ale nie może uruchomić plik.
- tryb dostępu dla grupy właścicieli „*r--*”, użytkowniki grupy właścicieli mogą czytać plik.
- tryb dostępu dla pozostałych użytkowników „*r--*”, wszystkie pozostałe użytkowniki mogą czytać plik.
- Właścicielem pliku jest użytkownik „*test*”
- Grupa właścicieli pliku jest „*users*”

Plik „*cachedrop.sh*”:

- flag: „-” - nie ma flagi.
- tryb dostępu dla właściciela: „*--x*” właściciel może tylko uruchomić plik.
- tryb dostępu dla grupy właścicieli „---”, użytkowniki grupy właścicieli nie mogą nic.
- tryb dostępu dla pozostałych użytkowników „---”, wszystkie pozostałe użytkowniki nie mogą nic.
- Właścicielem pliku jest użytkownik „*root*”
- Grupa właścicieli pliku jest „*root*”

Plik „*DevelopAppAndroid*”:

- flag: „*d*” - plik jest katalogiem.
- tryb dostępu dla właściciela: „*rwx*” właściciel może czytać, edytować, uruchomiać pliki.
- tryb dostępu dla grupy właścicieli „*rwx*”, użytkowniki grupy właścicieli mogą czytać, edytować, uruchomiać pliki.
- tryb dostępu dla pozostałych użytkowników „r-*x*”, wszystkie pozostałe użytkowniki mogą czytać i uruchomiać pliki, ale edycja plików zakazana.
- Właścicielem pliku jest użytkownik „*test*”
- Grupa właścicieli pliku jest „*user*”

Z tego wynika, że jeżeli na przykład użytkownik *bartek* z grupy *users* będzie próbował zedytować plik *750039.pdf* system na to nie pozwoli (*bartek* nie jest właścicielem, a użytkowniki grupy *users* mogą tylko czytać plik, a nie edytować).

Ważne rozumieć, że standardowy model DAC nie pozwala zbudować bezpieczny system. Implementacja DAC w systemie pozwala tylko na ograniczenie dostępu do zasobów ze strony nieupoważnionych użytkowników. W tym samym momencie aplikacja lub skrypt uruchomiony poprzez użytkownika odziedzicza uprawnienia dostępu użytkownika. Powoduje to problem bezpieczeństwa, ponieważ podatna na ataki aplikacja ma uprawnienia dostępu użytkownika który ją uruchomił.

Wyobraźmy sobie sytuację kiedy użytkownik korzysta z aplikacji VoIP, która jest podatna na ataki. Haker zaatakował aplikację VoIP i przejął kontrole nad tamtą aplikacją. Do czego haker ma dostęp w takim przypadku? Do tego do czego ma dostęp zhakowana aplikacja. A które uprawnienia ma zhakowana aplikacja? Aplikacja ma uprawnienia użytkownika który ją uruchomił. To znaczy, że haker, na przykład, ma dostęp do plików konfiguracji przeglądarki, do plików cookies, historii, do przechowanych w przeglądarce internetowej loginów i haseł i t.d.

Dla ochrony przed takimi przypadkami został opracowany model kontroli dostępu Mandatory Access Control (MAC), po polsku: obowiązkowa kontrola dostępu. Przykładami realizacji w systemach Linux są: AppArmor i SELinux. Zwykle, MAC kontroluje uprawnienia dostępu po sprawdzaniu systemu uprawnień DAC. To znaczy, że jeżeli system DAC odmawia w dostępie, to kontrola uprawnień systemem MAC nie zostanie uruchomiona. Jednak, jeżeli system DAC pozwala na dostęp do zasobów, możliwość takiego dostępu będzie sprawdzona poprzez MAC. To jest ważnym aspektem bezpieczeństwa systemu: MAC nie może pozwolić to co jest zakazane poprzez DAC, MAC może tylko dodatkowo ograniczyć to co jest dozwolone przez DAC. Z tego wynika, że błędna konfiguracja MAC w żaden sposób nie może redukować poziom bezpieczeństwa systemu OS (pod względem praw dostępu), ale może podnieść poziom bezpieczeństwa.

Jak, to działa? Na przykład: użytkownik uruchomił aplikację (przejętą przez hakera), która spróbuje odczytać pliki Firefox z hasłami i loginami. OS sprawdza poprzez DAC uprawnienia aplikacji na dostęp do tamtych plików. Aplikacja odziedzicza uprawnienia legalnego użytkownika, dla tego ma dostęp do plików. Ale dalsze kontrolą dostępu zajmie się AppArmor. Jeżeli w AppArmor jest założony profil dla tamtej aplikacji, to system sprawdzi czy ma tamta aplikacja dostęp do plików Firefox (nie użytkownik – a aplikacja). Wynikiem sprawdzania będzie odmowa dostępu dla aplikacji (jeżeli profil jest poprawnie skonfigurowany), niezależnie od tego który użytkownik ją uruchomił (nawet root).

W taki sposób poprawna konфiguracja systemu MAC, jest ważnym elementem zabezpieczenia OS, który pozwala wielokrotnie podnieść bezpieczeństwo systemu.

5. Wykrycie niesankcjonowanego dostępu.

Ważnym jest instalacja w systemie specjalnych programów dla wykrycia złośliwego kodu – programów antywirusowych. Niektóre programy antywirusowe dla OS Linux są zdolne do wykrycia nie tylko malwarów dla Linux ale i dla Windows. Przykładowym programem antywirusowym jest ClamAV.

Atakujący będzie próbować utrwalić się w systemie żeby uzyskać dostęp w dowolnym czasie bez ponownego hakowania. Takie próby, jeżeli odniosą sukces, zostawią ślady w systemie jako:

- anomalie logowania w systemie;
- anomalie w połączeniach sieciowych;
- anomalie w procesach systemowych;
- zmiany w plikach systemu.

Tamte anomalie wskazują również na nieudane próby ataku lub na to, że atak trwa w bieżącej chwile.

Dla tego warto sprawdzać i analizować pliki logów OS Linux, monitorować otwarte połączenia sieciowe, monitorować uruchomione procesy. Sprawdzać zmiany w plikach systemu. Dla wykrycia rootkitów i innych zmian w systemie służą takie pakiety jak: RKHunter, Tripwire, Aide, HIDS OSSEC.

Podsumowanie:

Warto rozumieć, że OS Linux może służyć dla budowania systemu o wysokim poziomie bezpieczeństwa i odporności na ataki, poziom bezpieczeństwa której jest znacznie wyższy od poziomu bezpieczeństwa większości OS. Elastyczność OS i dostępność całego szeregu narzędzi dla Linux pozwala na stworzenie unikalnego niestandardowego systemu zabezpieczenia, co znacznie utrudnia dla atakującego próby ataku. Jednak, domyślna instalacja systemu Linux nie może być traktowana jako bezpieczna i odporna na ataki.

Dla podniesienia poziomu bezpieczeństwa jest ważnym poprawna konfiguracja zapory sieciowej, sieciowych IDS+IPS, AppArmor lub SELinux, programu antywirusowego, HIDS (host-based intrusion detection system) i monitorowanie anomalii.