ZESZYT NAUKOWY
Wyższa Szkoła Zarządzania i Bankowości w Krakowie

Dr inż. Tomasz Wojdyński
Harol Hryc
Wydział Zarządzania, Finansów i Informatyki
Wyższa Szkoła Zarządzania i Bankowości w Krakowie

PROPOZYCJA ZABEZPIECZENIA SYSTEMU KOMPUTEROWEGO DLA SYTUACJI PO WYKORZYSTANIU LUK W OPROGRAMOWANIU DZIAŁAJĄCYM W LINUX

Wstęp

W epoce coraz szerszego dostępu do Internetu i coraz większego zapotrzebowania na usługi świadczone tą drogą, zwiększa się również ilość pojawiających się zagrożeń. Powoduje to, „że system komputerowy, który z grubsza można podzielić na cztery części: sprzęt, system operacyjny, programy użytkowe i użytkowników”[1], na którym są uruchomione świadczone usługi, musi tym zagrożeniom się przeciwstawić. Wielokrotnie można usłyszeć w mediach o włamaniach na strony internetowe, wycieku poufnych informacji czy uniemożliwieniu korzystania z zaatakowanego serwisu. Skutki takich działań mogą być katastrofalne. Dla przedsiębiorstw – upublicznienie danych może pociągnąć za sobą nawet upadłość (wykradzione technologie, szczegóły nowych produktów itp.), jak również sprawy sądowe założone przez poszkodowanych klientów (np. w przypadku wycieku danych klientów).[2] Zwykle winny takim zdarzeniom jest człowiek, natomiast zmienne jest miejsce, w którym został popełniony błąd, a tym samym różny może być „winowajca”. Począwszy od użytkownika, który w nieprawidłowy sposób obchodził się ze swoimi hasłami, przez administratorów, którym umknęły możliwe luki w zabezpieczeniach, a kolejno przez programistów, którym umknęły błędy w tworzonym przez nich oprogramowaniu, skończywszy na producentach sprzętu. To wszystko powoduje, że „do podstawowych zagadnień związanych z przechowywaniem i przetwarzaniem [przyp. a.] informacji w systemie komputerowych należy … chronienie jej … przed niewłaściwym dostępem (ochrona).”[3]

[1] A. Silberschatz, P.B. Galvin, *Podstawy systemów operacyjnych*, Wydawnictwo Naukowo-Techniczne, Warszawa 2002, str. 3.
[2] https://www.theguardian.com/technology/2015/aug/10/carphone-warehouse-uk-data-watchdog-investigating-customer-hack - data odczytu, 6.11.2016 r.
[3] Poz.cyt. A. Silberschatz, P.B. Galvin, *Podstawy …*, str. 422.

Niniejsza artykuł ma na celu zapoznanie czytelnika z najpopularniejszymi lukami w zabezpieczeniach systemu operacyjnego Linux oraz propozycjami zabezpieczenia się po pozytywnym skorzystaniu z nich przez intruza w celu zabezpieczenia systemu komputerowego przed sięgnięciem do zasobów innych niż skompromitowana usługa.

Punktem wyjścia będzie przedstawienie często występujących problemów z zabezpieczeniami, z którymi muszą sobie poradzić współczesne systemy (niekoniecznie tylko Linux). Koncepcje obrony przed wymienionymi niebezpieczeństwami zostaną jedna pominięte. Na początek nastąpiło omówienie problemów dotyczących haseł, które często bywają najsłabszym ogniwem w zabezpieczeniach systemu. Następnie przedstawiono problem związany z backdoorami i innymi furtkami, które zostały celowo napisane, aby osoby niepowołane mogły uzyskać dostęp do atakowanego systemu. Kolejno zostało wspomniane o błędach w aplikacjach, które nie zostały celowo zrobione, a mogą posłużyć jako wejście dla osób nieuprawnionych do systemu. Na końcu rozdziału jest mowa o zabezpieczeniach, które w przypadku kompromitacji uprzednio omówionych możliwych do użycia luk, zapobiegną lub utrudnią włamywaczowi dostęp do innych plików systemu.

1. Omówienie wybranych popularnych zagrożeń

Wraz z popularyzacją komputerów oraz spadającym poziomem wiedzy ich użytkowników, wzrasta również liczba zagrożeń z tego wynikająca. Głównie koncentrują się one wokół nieuprawnionych prób dostępu do systemów w celu przejęcia nad nimi kontroli, wykradaniu różnego rodzaju informacji, w celu ich bezpośredniego wykorzystania przez zleceniodawców ich nielegalnego pozyskania, czy to publikacji w celu kompromitacji podmiotów, których dotyczą. Listę takich nadużyć kończą okupy, kiedy to w zamian za wniesioną opłatę, pozyskane dane nie są ujawniane lub odwrotnie, dane, które zostały zablokowane/zaszyfrowane, zostają ponownie przywrócone prawowitym właścicielom.

1.1. Ataki oparte o próby pozyskania hasła

„Wszystkie konta użytkowników powinny posiadać hasła. Hasła odgrywają kluczową rolę w ogólnym zabezpieczeniu systemu”.[4] Często słabymi punktami w kwestiach zabezpieczeń systemów są hasła o niskiej złożoności, które mogą być w jakiś sposób złamane lub pozyskane.

[4] E. Frisch, *Unix – Administracja system – drugie wydanie,* Oficyna Wydawnicza READ ME, Łódź 1997, str. 156.

Potencjalnym źródłem wycieku, ułatwiającym pozyskanie hasła może być wielokrotne stosowanie tych samych haseł w różnych usługach. W tym przypadku rozwiązaniem jest stosowanie wszędzie unikalnych haseł. Jednak wtedy mogą wystąpić problemy z ich zapamiętaniem, gdyż w obecnych czasach ich liczba będzie duża z tendencją do ciągłego wzrostu. Ten problem można obecnie już z powodzeniem rozwiązać poprzez używanie schowków (menedżerów) na hasła (swego rodzaju baz danych przechowujących hasła, które są szyfrowane jednym – możliwie jak najsilniejszym hasłem głównym). Należy również pamiętać, by nie zapisywać nigdzie haseł, a jeżeli jest już taka konieczność, to należy zniszczyć nośnik z zapisanym hasłem, gdy przestaje on być potrzebny, a pod żadnym pozorem nie wyrzucać go do śmieci. Nigdy nie wiadomo czy ktoś nie będzie przeszukiwał np. śmieci celem zdobycia takich informacji.

Łamanie haseł jest inną metodą uzyskania niepowołanego dostępu, a jej szybkość zależy od konstrukcji samego hasła, jak również od systemu, do którego następuje włamanie. Jeżeli łamany jest *hash* hasła, który został uprzednio pozyskany przez np. podsłuchanie komunikacji między klientem a serwerem, wówczas czas potrzebny do jego złamania jest zależny od konstrukcji hasła oraz algorytmu obliczającego *hash* i mocy obliczeniowej sprzętu, na którym odbywa się jego łamanie. Podstawową metodą obrony, która jest wspólna dla obydwu wspomnianych wcześniej przypadków, jest stosowanie silnych haseł. Przy wyborze hasła powinno się zwracać uwagę aby hasło:

- nie zawierało znanych wyrazów – jest praktycznie pewne, że włamywacz będzie korzystał z ataków z wykorzystaniem słownika.
- Składało się z jak największej ilości klas znaków – zabezpiecza to przed atakami typu *brute-force* i pochodnymi (testowane są wszystkie kombinacje znaków).
- Było „odpowiedniej długości”.

Ostatecznie dobre hasło to ciąg losowych znaków typu: „}s)#]rjbHd5”; „_gmi7$D/:7n” czy „hkQJ9=kK4Z”. Długość natomiast powinna wynikać z tego jak często można próbować się zalogować podając nieprawidłowe hasło. Aby zrozumieć tę zależność przyjrzyjmy się takiemu oto przykładowi. Niech system umożliwia jedną próbę logowania na sekundę i hasło dostępu składa się z dużych i małych liter ASCII (2 * 26), cyfr (10) i znaków specjalnych (33) oraz zmiana takiego hasła niech odbywa się raz do roku. Teoretycznie w ciągu roku może odbyć się 31536000, 60 sekund razy 60 minut razy 24 godziny razy 365 dni co daje łącznie podaną wartość, prób złamania hasła. Dla tego przypadku już 4-znakowe hasło (81450625 możliwych kombinacji) powinno być odpowiednim. Tak jednak jest tylko w teorii. W praktyce można

szybko trafić na prawdziwe hasło w zależności od strategii jaką się przyjmie podczas próby odgadnięcia hasła, dlatego dobrze jest zwiększyć długość hasła o dodatkowe 2 znaki (co da łącznie ponad $7*10^{11}$ możliwych kombinacji). Jeżeli usługa, do której logujemy się za pomocą łamanego hasła, wykorzystuje mechanizmy, w których bardzo jest utrudnione pozyskanie *hashu* hasła, takie założenie odnośnie długości można uznać za rozsądne (np. połączenie wykorzystania protokołu *ssh* z uwierzytelnianiem przez system kerberos). Jeżeli jednak istnieje możliwość wycieku *hashu* hasła (np. niezabezpieczona strona internetowa z własnym mechanizmem logowania), to w przypadku popularnych algorytmów wyliczających hash, pozyskanie hasła to kwestia sekund – wykorzystując tablice tęczowe (uprzednio obliczone hashe haseł), lub jeżeli się tego zrobić nie da, ponieważ hasło było *solone* (do normalnego hasła dodano znany ciąg znaków), pozostaje łamanie *hashu*. Przykładowe prędkości (liczba prób na sekundę) dla popularnych *hashy* używając programu Hashcat v3.00-beta-145-g069634a na komputerze posiadającym osiem kart graficznych Nvidia GTX 1080 (karty graficzne całkiem dobrze się nadają do tego typu zadań, ponieważ zostały zaprojektowane do współbieżnego obliczania dużej ilości niezależnych od siebie danych – każda operacja obliczenia hashu jest niezależna od siebie, przez co można takie operacje zrównoleglić):

- MD5 200.3 GH/s;
- SHA1 68771.0 MH/s;
- SHA256 23012.1 MH/s;
- Whirlpool 2027.7 MH/s;
- phpass (używane w takich frameworkach jak Joomla czy Wordpress) 55130.8 kH/s;
- WPA/WPA2 3177.6 kH/s.[5]

Mając te wartości na uwadze i porównując z przykładowym 6-cio znakowym hasłem, którego hash byłby obliczony algorytmem MD5, sprawy wyglądają zatrważająco. Hasło zostanie pozyskane do 3 sekund! Jednak gdyby takie samo hasło zostało wykorzystane do zabezpieczenia sieci WiFi (WPA2), to jego pozyskanie trwało by do 64 godzin. Wartości te zakładają, że osoba łamiąca *hash* zna długość hasła. W przeciwnym przypadku ilość wszystkich kombinacji można wyrazić za pomocą wzoru $\sum_{i=1}^{n} a^i$, gdzie n oznacza długość hasła oraz a liczbę znaków w zestawie możliwych do użycia znaków (litery, cyfry, znaki specjalne).

Aby zmniejszyć ryzyko pozyskania hasła od strony systemowej, należy wydłużyć czas oczekiwania po nieudanej próbie logowania oraz zastosować mechanizm śledzący w logach

[5] https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40 - data odczytu, 6.11.2016 r.

próby logowania. Do tego należy dodać reguły odrzucające taki ruch jeżeli przekroczono dopuszczalny limit logowań w ciągu danego przedziału czasu (np. 20 nieudanych logowań w przeciągu godziny, skutkuje odrzucaniem połączeń z atakującego adresu na np. 3 godziny). W przypadku aplikacji, należy je często aktualizować, aby wykryte luki zostały załatane. Dodatkowo jeżeli jest się programistą takiej aplikacji, należy zwracać uwagę na zarządzanie hasłami w aplikacji: *hashe,* które powinny być solone i obliczane przy pomocy algorytmu o wysokiej złożoności obliczeniowej (np. SHA3 lub Whirlpool). Najlepiej zawsze używać systemu uwierzytelniania, w którym hasła nie są przechowywane na tym samym serwerze co aplikacja i nie przesyłają hashy przez sieć (np. kerberos). Ze strony użytkownika, należy pamiętać o tym, by nie używać jednego hasła do wielu serwisów, nie zapisywać haseł na karteczce (używać menadżera haseł, jeżeli są problemy z zapamiętaniem) i nie stosować haseł słownikowych (znanych wyrazów, zdań).

1.2. Backdoors

Backdoor to celowo napisany program lub część programu, który umożliwia niepowołany dostęp do systemu osobom trzecim. Tego typu szkodniki, w przypadku Linuksa, rozprzestrzeniają się najczęściej poprzez próbę łamania haseł SSH[6]. Inną możliwością jest pobranie spreparowanego obrazu instalacyjnego systemu z już zainstalowanym *backdoorem*. Kolejną możliwością jest uruchomienie spreparowanego pliku wykonywalnego, który instaluje złośliwy kod. Zdarza się również, że do niektórych programów zostanie dopisany *backdoor*. Przykładem takiej modyfikacji może być incydent z ProFTPD, gdzie nastąpiło włamanie na serwer dystrybucyjny i włamywacze zmodyfikowali kod źródłowy aplikacji. W efekcie ProFTPD w wersji 1.3.3c umożliwiał wykonywanie poleceń z uprawnieniami administratora, bez konieczności logowania się[7]. Innym ciekawym przypadkiem opisany w pozycji cytowanej Silberschatz A., Galvin P.B., *Podstawy systemów operacyjnych …* jest sytuacja gdzie: „Bardzo sprytne boczne wejście można zainstalować w kompilatorze. Kompilator może generować standardowy kod wynikowy oraz boczne wejście – niezależnie od kodu źródłowego, który kompiluje. Jest to działalność szczególnie perfidna, gdyż przeglądanie programu źródłowego nie uwidoczni żadnych niedociągnięć. Informację zawiera tylko kod źródłowy kompilatora.”[8]

[6] Przykład programu: http://vms.drweb-av.pl/virus/?_is=1&i=4323517 – data odczytu, 6.11.2016 r.
[7] https://www.aldeid.com/wiki/Exploits/proftpd-1.3.3c-backdoor – data odczytu 6.11.2016 r.
[8] Poz. cyt. Silberschatz A., Galvin P.B., *Podstawy …*, str. 746.

Podstawową możliwością zabezpieczenia się przed zagrożeniami tego typu jest nieinstalowanie oprogramowania z niepewnych źródeł oraz stosowanie silnych haseł. Dodatkowo powinno się zablokować możliwość zdalnego logowania się jako root. Niestety przed opisanym przypadkiem ataku na serwer dystrybutora oprogramowania nie jesteśmy w stanie się obronić inaczej niż poprzez obronę wewnętrzną, wykonaną przez samego dystrybutora. Z kolei jeżeli dana aplikacja ma udostępniać usługi przez sieć, dobrym pomysłem jest odizolowanie danej aplikacji od systemu poprzez użycie np. konteneryzacji. W przypadku braku takiej możliwości polecane będzie użycie mechanizmów „obowiązkowej kontroli dostępu" (MAC). Wbrew pozorom nie jest zalecane używanie funkcjonalności *chroot* jako mechanizmu ochrony systemu[9]. Jeżeli aplikacja wymaga uruchamiania z poziomu administratora, najbezpieczniejszym rozwiązaniem jest uruchomienie takiej aplikacji na osobnej maszynie, na której są tylko świadczone usługi tejże aplikacji. Należy o tym pamiętać i przestrzegać, gdyż „inaczej niż w innych systemach operacyjnych, superużytkownik [root, przyp. a.] posiada wszelkie prawa do systemu, w tym dostęp do wszystkich plików, poleceń itd. Z tej przyczyny >>nowy<< [przyp. a.] administrator w bardzo prosty sposób może ~~nieświadomie~~ [wykreślenie a.] spowodować bardzo poważną awarię systemu…".[10]

1.3. Błędy w aplikacjach

Jak pisze Steve McConnell, [tłum. a.]: „średnio na każde 1000 linii kodu pojawia się od ok. 1 do 25 błędów" [11]. W porównaniu do ilości linii kodu, jakie mają popularne programy, ilość potencjalnych błędów jest dość duża. Przykładowo serwer *http Apache* zawiera ponad 1,8 miliona linii kodu[12]. Biorąc pod uwagę powyższe założenie, program ten może mieć ok. 1 800 błędów. Pewna część tych błędów została zapewne naprawiona, jednak w projekcie tej wielkości ciężko (o ile w ogóle można) naprawić wszystkie błędy. Cześć z tych błędów może umożliwić dostęp osobie niepowołanej do informacji, do których nie powinna mieć dostępu. Ba!, nawet otwierać możliwość wykonywania różnych operacji z poziomem uprawnień dziurawego programu, łącznie z uruchomieniem powłoki.

Najpopularniejszym atakiem na aplikacje jest przepełnienie bufora. Stanowił on ok. 20% wszystkich ataków w 2005 roku[13] i pomimo, że taki atak jest coraz mniej skuteczny, zrozumienie go jest potrzebne do wyjaśnienia sposobu działania innego, dość groźnego ataku, przed

[9] https://access.redhat.com/blogs/766093/posts/1975883 – data odczytu 6.11.2016 r.
[10] Poz.cyt. E. Frisch, *Unix …*, str. 6.
[11] S. McConnell, *Code complete, 2nd edition*, Microsoft Press, Redmond 2004, rozdz. 22 str. 25.
[12] https://www.openhub.net/p/apache - data odczytu 6.11.2016 r.
[13] J. C. Foster, *Buffer Overflow Attacks: Detect, Exploit, Prevent,* Syngress Publishing, Rockland 2005, str. 20.

którym nie ma w pełni skutecznego zabezpieczenia. Zasada działania zastosowana w tym ataku polega najpierw na analizie programu, czy wszystkie możliwe dojścia (miejsca do wprowadzania danych) są zabezpieczone przed nadmierną ich ilością. Jeżeli zostanie wprowadzone zbyt dużo danych, jest szansa, że zostaną nadpisane inne zmienne w programie, łącznie z adresem powrotu funkcji. Poznając taki błąd można go wykorzystać do wykonania kodu, który wprowadził atakujący. Zwykle ma on na celu uruchomienie powłoki (Shellcode). Współcześnie tego typu atak jest mocno utrudniony z powodu ustawienia w programach wszystkich sekcji, w których dane mogą się zmieniać jako niewykonywalne, natomiast sekcja z uruchamianym kodem jest ustawiona w trybie tylko do odczytu. Implementacja tego mechanizmu jest zależna od procesora (NX bit) i w starszych procesorach bez tej funkcjonalności potrzebna jest jej emulacja.

O wiele trudniejszym do wykonania atakiem jest tzw. *return-to-libc*, który obchodzi bit NX, co powoduje, że jest on również trudny do zabezpieczenia. Został on dokładnie opisany w pozycji Defeating Solaris/SPARC non-executable stack protection[14] w 1999 roku. J. McDonald opublikował kod wykonujący ten typ ataku na systemie Solaris 2.6, działający na architekturze SPARC. Co więcej, w roku 2007 powstała publikacja, która zawiera [tłum. a.]: „inne godne uwagi przykłady takich *exploit'ów* na tę samą architekturę, począwszy od prostych ataków wykorzystujących tę technikę, aż do technik hybrydowych wstrzykiwania złośliwego kodu do wykonywlanych segmentów pamięci atakowanego programu".[15] Polega on na tym, iż wykorzystuje się już istniejący kod, który jest wykonywalny, najczęściej standardowej biblioteki języka C (stąd nazwa *return-to-libc*), ponieważ prawie wszystkie programy z niej korzystają. Trudność polega na tym, aby znaleźć adres interesującej funkcji (np. system(), która służy do wykonania polecenia będącego jej argumentem) oraz argumentu, najczęściej ciągu znaku „/bin/sh". Sposobem utrudnienia tego ataku jest głównie użycie techniki *Address space layout randomization*, która polega na tym, że za każdym razem kiedy uruchamiany jest program, adresy funkcji i inne ważne struktury wraz z funkcjami z bibliotek, z których korzysta, zostają losowo zmienione co utrudnia trafienie w potrzebne do wykonania tego ataku zasoby.

Podstawowym zabezpieczeniem przed tego typu atakami jest pisanie bezpiecznego kodu, co powoduje, że głównie osobą odpowiedzialną za bezpieczeństwo jest programista aplikacji. Jego odpowiedzialnością jest sprawdzanie, czy wprowadzane dane są poprawne i w odpowiedniej ilość. Można również użyć narzędzi w postaci *Valgrind* w celach pomocy w debugowaniu.

[14] J. McDonald, *Defeating Solaris/SPARC non-executable stack protection*. Bugtraq, Mar. 1999, Online: http://seclists.org/bugtraq/1999/Mar/4.
[15] M. Ivaldi, *Re: Older SPARC return-into-libc exploits. Penetration Testing*, Aug. 2007.

Ze strony administratora zabezpieczanego systemu, w którym ma działać aplikacja, można zastosować techniki oznaczające możliwość wykonywania kodu czy randomizację adresów zasobów programu, jednak nie rozwiąże to każdego problemu.

1.4. Zabezpieczenie przed dostępem do danych w przypadku kompromitacji usługi

Kiedy włamywaczowi uda się przejąć kontrolę nad jakąś usługą (np. wywołanie powłoki atakiem *return-to-libc*), wówczas ma on dostęp do wszelkich danych, z których dana usługa korzysta. Przykładowo jeżeli ofiarą jest serwer WWW i używany jest protokół https, włamywacz może uzyskać dostęp do certyfikatów szyfrowania i nic go przed tym nie powstrzyma, bowiem działa z takimi uprawnieniami z jakimi działa usługa, którą skompromitował.

Można się zabezpieczyć w taki sposób, aby włamywacz nie mógł uzyskać dostępu to tych części systemu, które nie są potrzebne aby program działał. Dość skutecznym sposobem jest zablokowanie jakiejś usłudze możliwości wykonywania jakichkolwiek programów, przez co zablokuje się możliwość uruchomienia powłoki w skompromitowanej aplikacji, utrudniając pole manewru włamywaczowi.

2. Praktyczne zastosowanie rozwiązań obrony po skompromitowaniu aplikacji

Kolejny rozdział koncentruje się na konfiguracji systemu pod kątem sytuacji, w której włamywacz skompromitował jakąś aplikację. Zaproponowane zostanie w nim przejście ze standardowego modelu kontroli dostępu DAC na MAC lub RBAC. Omówiony będzie proces instalacji i konfiguracji odpowiednich pakietów.

2.1. Wstępna konfiguracja systemów

Do praktycznych rozważań nad zabezpieczaniem systemów przed wykorzystaniem znanych luk bezpieczeństwa w systemach linuksowych wybrano trzy dystrybucje: CentOS, Debian oraz Gentoo.

Dla systemu CentOS wybrano instalacja systemu z obrazu „CentOS-7-x86_64-Minimal-1511”. Podczas instalacji wybrano profil bezpieczeństwa „Default”.

Dla systemu Debian instalacja systemu została wykonana z obrazu „debian-8.6.0-amd64-netinst”. Na liście „Wybór oprogramowania” wybrano tylko pole „Podstawowe narzędzia systemowe”, wszystko inne zostało odznaczone.

Instalację systemu Gentoo wykonano wykorzystując obraz install-amd64-minimal-20160929 oraz obraz systemu stage3-amd64-20160929. Proces instalacji tego systemu zostanie nieco bardziej przybliżony z powodu różnych możliwości konfiguracyjnych, na jakie napotyka

się podczas tego procesu, a które mają wpły na końcowy efekt osiągnięcia bezpiecznej konfiguracji. Ostatecznie zostanie wybrany profil „Hardened", który włącza opcje poprawiające bezpieczeństwo systemu. I od tego właśnie momentu przedstawiony zostanie proces instalacji. Zaczynamy więc od przygotowanie systemu Portage do instalacji oprogramowania, gdyż początkowo zainstalowany wstępnie system nie posiada repozytorium, należy więc je pobrać:

```
livecd gentoo_root # emerge –sync
```

Następnie potrzebna jest zmiana profilu systemu. Listę dostępnych profili można uzyskać wpisując:

```
livecd / # eselect profile list
Available profile symlink targets:
[1]   default/linux/amd64/13.0
[2]   default/linux/amd64/13.0/selinux
[3]   default/linux/amd64/13.0/desktop
[4]   default/linux/amd64/13.0/desktop/gnome
[5]   default/linux/amd64/13.0/desktop/gnome/systemd
[6]   default/linux/amd64/13.0/desktop/kde
[7]   default/linux/amd64/13.0/desktop/kde/systemd
[8]   default/linux/amd64/13.0/desktop/plasma
[9]   default/linux/amd64/13.0/desktop/plasma/systemd
[10]  default/linux/amd64/13.0/developer
[11]  default/linux/amd64/13.0/no-multilib *
[12]  default/linux/amd64/13.0/systemd
[13]  default/linux/amd64/13.0/x32
[14]  hardened/linux/amd64
[15]  hardened/linux/amd64/selinux
[16]  hardened/linux/amd64/no-multilib
[17]  hardened/linux/amd64/no-multilib/selinux
[18]  hardened/linux/amd64/x32
[19]  hardened/linux/musl/amd64
[20]  hardened/linux/musl/amd64/x32
[21]  default/linux/uclibc/amd64
[22]  hardened/linux/uclibc/amd64
livecd / #
```

Jak widać, wybór jest duży. Na potrzeby tego artykułu zostanie wybrana opcja 16 – hardened bez możliwości uruchamiania aplikacji 32bitowych i bez SELinux, którego funkcje zostaną przedstawione na przykładzie dystrybucji CentOS. Ma to na celu ukazanie alternatywnych rozwiązań w porównaniu do innych dystrybucji.

```
livecd / # eselect profile set 16
livecd / #
```

Zmiana profilu na hardened wprowadza modyfikacje opcji niektórych programów. Początkowo należy rekompilować kompilator, aby mógł budować oprogramowanie z włączonymi mechanizmami poprawiającymi bezpieczeństwo oraz pakiet binutils.

```
livecd / # emerge gcc
livecd / # emerge binutils
```

Jako jądro zostanie wykorzystana wersja z dodanymi łatkami zwiększającymi bezpieczeństwo:

```
livecd / # emerge hardened-sources
```

W dniu pisania tego tekstu, aktualną stabilną wersją tego jądra była 4.4.8-r1. Przechodzimy do konfiguracji jądra. Domyślnie źródła jąder są instalowane w katalogu /usr/src/.

```
livecd / # cd /usr/src/linux-4.4.8-hardened-r1/
```

Zostanie teraz omówiony proces konfiguracji, jednak główna uwaga zostanie poświęcona konfiguracji opcji bezpieczeństwa.

```
livecd linux-4.4.8-hardened-r1 # make menuconfig
```

Na początek zostaną wybrane standardowe mechanizmy zwiększające bezpieczeństwo jakie są w podstawowej wersji jądra:

General setup --->
Stack Protector buffer overflow detection (None) --->

Należy zmienić z None na Strong.

[ ] Disable heap randomization

Należy upewnić się, że opcja jest wyłączona.

Processor type and features --->
[ ] Intel MPX (Memory Protection Extensions)

Jeżeli system będzie działał na procesorach z obsługą rozszerzeń MPX (na chwilę pisania artykułu, jedyne procesory z tym rozszerzeniem to te oparte o architekturę Intel Skylake) należy zaznaczyć tę opcję:

vsyscall table for legacy applications (Emulate) --->

Należy zmienić z Emulate na None.

[*] Allow userspace to modify the LDT by default

Tę pozycję należy wyłączyć, pomimo że może powodować problemy z działaniem takich programów jak Dosemu lub Wine, ale wyłączenie to znacząco zwiększa bezpieczeństwo systemu.

Poniższe opcje dotyczą tylko jąder z łatką *grsecurity* (hardened-sources takimi są) i konfiguracja znajduje się w **Security options --->Grsecurity --->[ ] Grsecurity**

Po zaznaczeniu pola **[ ] Grsecurity** rozwinie się menu:

```
[*] Grsecurity
    Configuration Method (Custom) --->
    Customize Configuration --->
```

Dla ułatwienia wyboru opcji konfiguracyjnych (których jest dość obszerny wybór) zaleca się zmianę **Configuration Method** z *Custom* na *Automatic*. Skutkuje to pojawieniem się dodatkowych opcji:

```
[*] Grsecurity
    Configuration Method (Automatic) --->
    Usage Type (Server) --->
    Virtualization Type (None) --->
    Required Priorities (Performance) --->
    Default Special Groups --->
    Customize Configuration --->
    Usage Type (Server) --->
```

Ostatnią opcję należy zostawić jako Server, chyba że system ma działać w roli końcówki klienckiej.

```
Virtualization Type (None) --->
```

Jeżeli system działa na maszynie fizycznej i nie są na nim uruchamiane maszyny wirtualne (nie jest hostem wirtualizacyjnym), tą opcję należy zostawić na *None*. Jeśli jednak ma pełnić rolę hosta do wirtualizacji, należy zmienić na *Host*. W przypadku omawianym system jest gościem, więc typ zostanie zmieniony na *Guest*. Jeżeli jest wybrana inna opcja niż *None*, pojawią się poniżej dodatkowe opcje:

```
Virtualization Hardware (EPT/RVI Processor Support) --->
```

Tą należy nostawić na wartości domyślnej (EPT/RVI Processor Support) jeżeli procesor, na którym działa system, posiada i ma włączone rozszerzenia wirtualizacyjne: w przypadku procesorów Intela – EPT (Extended Page Tables), w przypadku AMD – RVI (Rapid Virtualization Indexing). Pierwsze procesory jakie posiadały takie możliwości, to architektura Intel

Nehalem i AMD Barcelona. Można sprawdzić czy procesor naszego systemu komputerowego wspiera te rozszerzenia poleceniem: *grep " ept\|rvi" /proc/cpuinfo* Jeżeli cokolwiek się wyświetli, oznacza to wsparcie dla tych rozszerzeń, w przeciwnym wypadku należy wybrać opcję:

First-gen/No Hardware Virtualization
Virtualization Software (Xen) --->

Drugą dodatkową opcją, która pojawi się dodatkowo, jeśli wybierzemy powyżej inne ustawienie niż *None*, jest wybór oprogramowania do wirtualizacji. Dla potrzeb tego artykułu wybrany zostanie VMWare:

Required Priorities (Performance) --->

Celem tej konfiguracji jest uzyskanie możliwie największego poziomu bezpieczeństwa instalowanego systemu, dlatego należy wybrać opcję *Security* zamiast *Performance*. Warto jednak pamiętać o tym, że autorzy łatki podają, iż w najgorszym przypadku wydajność spowodowana przez ten wybór może się zmniejszyć nawet o 20%.

Opcjonalnie można wyłączyć SELinux, gdyż w tej konfiguracji nie będzie on używany (zamiast niego użyty zostanie RBAC):

Security options --->[*] NSA SELinux Support

Teraz można skompilować jądro i je zainstalować:

```
livecd linux-4.4.8-hardened-r1 # make -j8
```

Jeżeli system ma inną liczbę wątków niż 8, zmienić na opcję make'a na -jX gdzie X = liczba wątków. Po skompilowaniu należy zainstalować jądro oraz jego moduły:

```
livecd linux-4.4.8-hardened-r1 # make install
livecd linux-4.4.8-hardened-r1 # make modules_install
```

Kolejno wykonujemy rekompilację biblioteki glibc:

```
livecd / # emerge glibc
```

oraz rekompilacja całego systemu:

```
livecd / # emerge --update –oneshot --emptytree --deep --with-bdeps\=y --newuse --changed-use @system @world
```

Pozostaje instalacja bootloadera:

```
livecd / # emerge grub
livecd / # grub-install /dev/sda
```

i wygenerowanie konfiguracji bootloadera:

```
livecd / # grub-mkconfig -o /boot/grub/grub.cfg
```

Upewniając się, że mamy ustawione dopowiednie hasło dla root wykonujemy reboot systemu:

```
livecd ~ # reboot
```

który powinien być już gotowy do dalszych czynności konfiguracyjnych.

2.2. Obrona przed dostępem do danych w przypadku kompromitacji usługi

Przedstawione w tej części rozwiązania można traktować jako „środki po", czyli takie, które jedynie zapobiegną dostępowi do danych, z których normalnie aplikacja nie korzysta.

2.2.1. SELinux

SELinux czyli Security-Enhanced Linux jest implementacją systemu obowiązkowej kontroli dostępu (MAC), zapoczątkowanej przez NSA, a aktualnie głównie rozwijaną przez firmę Redhat. W związu z tym w dystrybucjach Linuxa tej gałęzi znajdziemy najlepiej działający SELinux z dobrze zdefiniowanymi politykami do praktycznie każdego pakietu wchodzącego w skład standardowego repozytorium. Zatem do prezentacji SELinux zostanie wykorzystany CentOS, w którym jest on domyślnie jest działający i prawidłowo skonfigurowany.

W tej części artykułu zostanie ogólnie omówiony SELinux oraz zostanie pokazane w jaki sposób go używać. Tworzeniem polityk zajmują się głównie twórcy dystrybucji i zostanie ten temat ze względu na jego złożoność pominięty.

SELinux działa głównie w oparciu o konteksty, które jest przechowywany w etykietach bezpieczeństwa systemu plików (*security labels*) oraz o polityki, które określają uprawnienia między podmiotami opatrzonymi takimi kontekstami. Przykładowo polityka pozwala pewnemu procesowi na odczyt i wykonywanie pliku oznaczonego kontekstem „A" a na zapis i odczyt pliku oznaczonego kontekstem „B", natomiast dostęp do wszystkich innych plików oznaczonymi innymi kontekstami jest zabroniony. W celu uzyskania informacji o kontekście należy dodać opcję *-Z* do polecenia *ls*. Przykładowy kontekst wygląda jak poniżej:

```
[root@CentOS7 ~]# ls -Z /bin/bash
-rwxr-xr-x. root root system_u:object_r:shell_exec_t:s0 /bin/bash
[root@CentOS7 ~]#
```

Poza uprawnieniami do pliku i informacji o właścicielu, pojawił się następujący ciąg znaków: „*system_u:object_r:shell_exec_t:s0*". Ciąg ten jest podzielony dwukropkami na następujące pola:

- ***system_u*** – określa użytkownika SELinux
- ***object_r*** – określa rolę SELinux

- ***shell_exec_t*** – określa typ (w przypadku procesu, zwane jest również jako *domena procesu*) SELinux
- ***s0*** – określa wrażliwość

W większości przypadków, najistotniejszym polem jest typ/domena, ponieważ ponad 99 procent reguł operuje na relacjach między typami/domenami, nawet nie wspominając o pozostałych polach.[16] W tym przypadku powłoka bash ma specjalny typ przypisywany plikom wykonywalnym powłoki. Jest to celowy zabieg służący zapobieganiu uruchamiania programom powłoki na wypadek przejęcia przez włamywacza kontroli nad programem (zwykle celem włamania jest uruchomienie powłoki na docelowym systemie).

Poza kontekstami przypisanymi plikom na dysku jest jeszcze kontekst użytkownika. Sprawdzić kontekst aktualnie zalogowanego użytkownika można w następujący sposób:

```
[root@CentOS7 ~]# id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@CentOS7 ~]#
```

W tym przypadku użytkownik jest w domenie *unconfined* – specjalnej domenie, która praktycznie pozwala na wszystko. Przeważnie jest to oznaką, że SELinux ma aktywną politykę „*targeted*" (nazwy mogą być różne w zależności od dystrybucji). Celem tej polityki jest zabezpieczanie systemu od strony procesów działających w tle (stad nazwa – polityka jest „wycelowana" w konkretne programy/usługi). Więcej informacji można uzyskać poprzez narzędzie *sestatus:*

```
[root@CentOS7 ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
[root@CentOS7 ~]#
```

Można się z przedstawionych informacji dowiedzieć, że SELinux działa z polityką „*targeted*". Aby uzyskać informacje dotyczące domen w jakich działają procesy w systemie można posłużyć się poleceniem *ps* z opcją *Z*. Zademonstrowany zostanie inny sposób wypisywania procesów, aby był lepiej czytelny:

[16] S. Vermeulen, *SELinux System Administration*, Packt Publishing 2013, s. 26.

```
[root@CentOS7 ~]# ps -o user,pid,ucomm,context -A | grep -v kernel_t
USER       PID COMMAND         CONTEXT
root         1 systemd         system_u:system_r:init_t:s0
root       740 systemd-journal system_u:system_r:syslogd_t:s0
root       759 lvmetad         system_u:system_r:lvm_t:s0
root       875 auditd          system_u:system_r:auditd_t:s0
root       895 systemd-logind  system_u:system_r:systemd_logind_t:s0
root       896 rsyslogd        system_u:system_r:syslogd_t:s0
root       897 NetworkManager  system_u:system_r:NetworkManager_t:s0
root       898 vmtoolsd        system_u:system_r:vmtools_t:s0
dbus       899 dbus-daemon     system_u:system_r:system_dbusd_t:s0-s0:c0.c1023
root       905 irqbalance      system_u:system_r:irqbalance_t:s0
root       929 wpa_supplicant  system_u:system_r:NetworkManager_t:s0
polkitd    930 polkitd         system_u:system_r:policykit_t:s0
root      1141 sshd            system_u:system_r:sshd_t:s0-s0:c0.c1023
root      1142 tuned           system_u:system_r:tuned_t:s0
root      1220 master          system_u:system_r:postfix_master_t:s0
postfix   1222 qmgr            system_u:system_r:postfix_qmgr_t:s0
root      1279 agetty          system_u:system_r:getty_t:s0-s0:c0.c1023
root      1703 crond           system_u:system_r:crond_t:s0-s0:c0.c1023
root      1738 systemd-udevd   system_u:system_r:udev_t:s0-s0:c0.c1023
root     12549 dhclient        system_u:system_r:dhcpc_t:s0
root     12579 sshd            unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
root     12583 bash            unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
postfix  14331 pickup          system_u:system_r:postfix_pickup_t:s0
root     14403 ps              unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
root     14404 grep            unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@CentOS7 ~]#
```

Powyższa komenda wyświetla dla wszystkich procesów: nazwę użytkownika, z którego uprawnieniami działa proces; identyfikator procesu; nazwę procesu; domenę procesu. Warto zauważyć, że jedynie programy użytkownika działają w domenie *unconfined* a wszystkie inne w indywidualnie dla nich przygotowanych domenach.

Poniżej zostanie przedstawione praktyczne zastosowanie SELinux wraz z typowym problemem na jaki można się natknąć.

Udostępnianie plików przez http: W systemie jest zainstalowany serwer http (w tym przypadku apache) oraz firewall jest odpowiednio skonfigurowany (przyjmowanie połączeń na porcie 80/tcp). Sam serwer ma jednak zmieniony katalog ze stronami (DocumentRoot) z domyślnego /var/www/html na /mywebsite (katalog nie jest jeszcze utworzony). Użytkownik chce stworzyć odpowiedni katalog, umieścić w nim plik i pobrać go na zdalnym komputerze:

```
[root@CentOS7 ~]# mkdir /mywebsite
[root@CentOS7 ~]# cd /mywebsite/
[root@CentOS7 mywebsite]# echo "test strony" > index.html
[root@CentOS7 mywebsite]# ls -al
razem 8
```

```
drwxr-xr-x.  2 root root   23 12-18 01:44 .
dr-xr-xr-x. 18 root root 4096 12-18 01:43 ..
-rw-r--r--.  1 root root   12 12-18 01:44 index.html
[root@CentOS7 mywebsite]# systemctl start httpd
```

Tymczasem na zdalnym komputerze:

```
user@client /tmp $ wget http://172.21.6.10/
--2016-12-18 01:50:53--  http://172.21.6.10/
Łączenie się z 172.21.6.10:80... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 403 Forbidden
2016-12-18 01:50:53 BŁĄD 403: Forbidden.
user@client /tmp $
```

Z jakiegoś powodu nie można uzyskać dostępu do pliku. Analizując logi można natrafić na takie informacje:

```
[root@CentOS7 ~]# tail -n 1 /var/log/httpd/error_log
[Sun Dec 18 01:50:53.062055 2016] [core:error] [pid 15345] (13)Permission denied: [cli-
ent172.21.6.33:41910] AH00035: access to /index.html denied (filesystem path '/mywebsite
/index.html') because search permissions are missing on a component of the path
[root@CentOS7 ~]#
```

Jednakże uprawnienia pliku i katalogu pozwalają każdemu na odczytanie zawartości pliku i wejście do katalogu. W tym przypadku będzie pomocny *audit.log*, w którym zapisywane są informacje związane z SELinux:

```
[root@CentOS7 ~]# tail -n 2 /var/log/audit/audit.log
type=AVC msg=audit(1482022253.061:259): avc:  denied  { getattr } for  pid=15345
comm="httpd" path="/mywebsite/index.html" dev="dm-0" ino=50617843 scontext=sys-
tem_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0 tclass=file
type=SYSCALL msg=audit(1482022253.061:259): arch=c000003e syscall=6 success=no
exit=-13 a0=7efcbcff8118 a1=7ffc88838b90 a2=7ffc88838b90 a3=1 items=0 ppid=15343
pid=15345 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48
fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd" subj=sys-
tem_u:system_r:httpd_t:s0 key=(null)
[root@CentOS7 ~]#
```

Zwykle wpisy typu AVC i SYSCALL pojawiają się razem, jednak bardziej przydatny jest typ AVC, z którego można się dowiedzieć że: została zablokowana operacja *getattr* (nim plik zostanie odczytany, uprzednio muszą zostać odczytane jego atrybuty, stąd *getattr*) dla procesu o identyfikatorze 15345 i nazwie httpd dla ścieżki /mywebsite/index.html na urządzeniu dm-0 (dysk twardy), której i-węzeł (i-node) ma identyfikator 50617843, gdzie kontekst procesu chcącego uzyskać dostęp do pliku był system_u:system_r:httpd_t:s0 a kontekst zasobu był unconfined_u:object_r:default_t:s0 oraz zasób był plikiem.

Podejrzewając, że to SELinux zabrania dostępu, można go tymczasowo przestawić w tryb permissive, który wyłącza jakiekolwiek blokowanie dostępu (jednak zdarzenia wciąż są logowane):

```
[root@CentOS7 ~]# setenforce 0
[root@CentOS7 ~]# getenforce
Permissive
[root@CentOS7 ~]#
```

Poleceniem getenforce można sprawdzić w jakim trybie jest SELinux. Teraz można jeszcze raz spróbować pobrać plik:

```
user@client /tmp $ wget http://172.21.6.10/
--2016-12-18 02:26:04--  http://172.21.6.10/
Łączenie się z 172.21.6.10:80... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
Długość: 12 [text/html]
Zapis do: `index.html'

index.html                  100%[=================================>]      12  --
.-KB/s    in 0s

2016-12-18 02:26:04 (1,31 MB/s) - zapisano `index.html' [12/12]
user@client /tmp $ cat index.html
test strony
user@client /tmp $
```

Przyczyna znaleziona, jednak pozostawianie SELinux w trybie permissive mija się z celem. Rozwiązaniem problemu jest ustawienie odpowiedniego kontekstu, lecz nie zawsze jest wiadome jaki kontekst jest dozwolony. Z pomocą przychodzi narzędzie sesearch (domyślnie w CentOS7 nie jest ono zainstalowane. Można je zainstalować poleceniem yum install setools-console). Potrzebny szablon zapytania wygląda następująco:

```
sesearch -s <kontekst źródłowy> -c <klasa obiektu docelowego> -p <pożądana operacja> -
Ad.[17]
```

Jako kontekst źródłowy zostanie użyty typ z scontext z logów AVC, w tym przypadku *httpd_t*. Klasą obiektu będzie tclass, czyli *file*. Pożądaną operacją wbrew pozorom nie jest *getattr* a *read*. Zwykle na pliku można wykonywać następujące operacje: *ioctl read write create getattr setattr lock append unlink link rename open execute execute_no_trans entrypoint.*

```
root@CentOS7 ~]# sesearch -s httpd_t -c file -p read -Ad | head -n 1
Found 220 semantic av rules:
[root@CentOS7 ~]#
```

[17] https://wiki.gentoo.org/wiki/SELinux/Tutorials/How_does_a_process_get_into_a_certain_context - data odczytu 18.12.2016 r.

220 pasujących reguł nie napawa optymizmem. Można to jednak zawęzić – używając polecenia grep można pozbyć się wpisów z operacjami niechcianymi (takimi jak write czy execute):

```
[root@CentOS7 ~]# sesearch -s httpd_t -c file -p read -Ad | grep -v execute | grep -v write |
wc -l
91
[root@CentOS7 ~]#
```

Jednocześnie można zaobserwować pojawianie się w nazwach słowa „content", co jest odpowiednie dla potrzeb hostingu plików:

```
[root@CentOS7 ~]# sesearch -s httpd_t -c file -p read -Ad | grep -v execute | grep -v write |
grep content | grep user
  allow httpd_t httpd_user_ra_content_t : file { ioctl read getattr lock open } ;
  allow httpd_t httpd_user_content_type : file { ioctl read getattr lock open } ;
[root@CentOS7 ~]#
```

Z wyświetloną listą jest jeszcze jeden problem. Przedstawia ona wszystkie konteksty, a inny kontekst jest dla procesu i inny dla pliku. W tym przypadku pomocne będzie polecenie seinfo -afile_type -x, które wyświetla wszystkie konteksty jakie można przypisać plikom. Dzięki wiedzy uzyskanej na podtawie poprzedniego polecenia można - używając polecenia grep - odfiltrować niechciane wyniki:

```
[root@CentOS7 ~]# seinfo -afile_type -x | wc -l
2881
[root@CentOS7 ~]# seinfo -afile_type -x | grep httpd | grep content | grep user
    httpd_user_content_t
    httpd_user_rw_content_t
    httpd_user_ra_content_t
[root@CentOS7 ~]#
```

Z przedstawionych powyżej, najbardziej pasującym typem jest: httpd_user_content_t. Można więc spróbować zmienić kontekst katalogu i pliku na powyższy a następnie wystarczy przestawić SELinux na tryb enforcing i ponownie spróbować pobrać plik:

```
[root@CentOS7 ~]# ls -alZ /mywebsite/
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 .
dr-xr-xr-x. root root system_u:object_r:root_t:s0      ..
-rw-r--r--. root root unconfined_u:object_r:default_t:s0 index.html
[root@CentOS7 ~]# chcon -R -t httpd_user_content_t /mywebsite/
[root@CentOS7 ~]# ls -alZ /mywebsite/
drwxr-xr-x. root root unconfined_u:object_r:httpd_user_content_t:s0 .
dr-xr-xr-x. root root system_u:object_r:root_t:s0      ..
-rw-r--r--. root root unconfined_u:object_r:httpd_user_content_t:s0 index.html
[root@CentOS7 ~]# setenforce 1
```

```
[root@CentOS7 ~]# getenforce
Enforcing
[root@CentOS7 ~]#
```

Dodatkowo należy ustawić flagę w selinux, która umożliwia dostęp procesowi *httpd* do plików z kontekstem *httpd_user_content_t*. W celu wypisania wszystkich flag należy posłużyć się poleceniem *getsebool -a*. Lista flag dotyczących procesu httpd jest długa i od razu przejdziemy do tych najbardziej interesujących, tzn. zawierających wpisy httpd oraz user:

```
[root@CentOS7 ~]# getsebool -a | grep httpd | grep user
httpd_read_user_content --> off
[root@CentOS7 ~]#
```

Włączenie jej, używając polecenia *setsebool -P* rozwiąże problem z odmową dostępu:

```
[root@CentOS7 ~]# setsebool -P httpd_read_user_content 1
[root@CentOS7 ~]# getsebool -a | grep httpd | grep user
httpd_read_user_content --> on
[root@CentOS7 ~]#
```

Można teraz przetestować wprowadzone zmiany:

```
user@client /tmp $ wget http://172.21.6.10/
--2016-12-18 03:20:05--  http://172.21.6.10/
Łączenie się z 172.21.6.10:80... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
Długość: 12 [text/html]
Zapis do: `index.html'

index.html                  100%[=================================>]      12  --
.-KB/s    in 0s

2016-12-18 03:20:05 (1,31 MB/s) - zapisano `index.html' [12/12]
user@client /tmp $
```

Problem rozwiązany. Od teraz wszystkie nowo tworzone pliki w tym katalogu będą dziedziczyły do niego kontekst, przez co nie ma potrzeby przypisywania do nich prawidłowego kontekstu.

Jest to tak naprawdę tylko wprowadzenie do używania SELinux. Po więcej informacji, autorzy odsyłają do książki Vermeulen S., SELinux System Administration, Packt Publishing 2013.

2.2.2. RBAC

RBAC (*Role-based access controll*) jest podsystemem w *grsecurity*, który pełni rolę systemu obowiązkowej kontroli dostępu. W porównaniu do SELinux'a, RBAC jest o wiele prostszy w użytkowaniu. Nie wymaga systemu plików z rozszerzonymi atrybutami (*xattrs*) i listy

dostępu generują się automatycznie poprzez mechanizm samouczenia. Jedną z niewielu dystrybucji, która oficjalnie wspiera to rozwiązanie, jest Gentoo, dlatego zostanie na nim przeprowadzona jego demonstracja (SELinux również jest dostępny na tym systemie).

Domyślnie RBAC jest wyłączony oraz narzędzie do administracji nie jest zainstalowane. Należy więc je zainstalować:

```
gentoo ~ # emerge gradm
```

Po instalacji wymagane jest ustawienie trzech haseł:

```
gentoo ~ # gradm -P
Setting up grsecurity RBAC password
Password:
Re-enter Password:
Password written to /etc/grsec/pw.
gentoo ~ # gradm -P admin
Setting up password for role admin
Password:
Re-enter Password:
Password written to /etc/grsec/pw.
gentoo ~ # gradm -P shutdown
Setting up password for role shutdown
Password:
Re-enter Password:
Password written to /etc/grsec/pw.
gentoo ~ #
```

Można teraz sprawdzić czy RBAC się uruchamia poprzez polecenie:

```
gentoo ~ # gradm -E
Warning: permission for symlink /dev/cdrom in role default, subject / does not match that of
its matching target object /dev.  Symlink is specified on line 332 of /etc/grsec/policy.
gentoo ~ #
```

Powyższe ostrzeżenie można zignorować, ponieważ dotyczy standardowej polityki, z którą jest instalowany *gradm*, a ta zostanie zamieniona. Następnym krokiem jest wygenerowanie polityki poprzez uczenie, jednak najpierw należy wyłączyć RBAC:

```
entoo ~ # gradm -D
Password:
gentoo ~ #
```

Hasłem jest główne hasło RBAC (w demonstracji wprowadzane jako pierwsze „Setting up grsecurity RBAC password"). W tym momencie zalecane jest zainstalowanie docelowego oprogramowania używanego w systemie. Po ewentualnej instalacji można rozpocząć rejestracje zdarzeń w systemie:

```
gentoo ~ # gradm -F -L /etc/grsec/learning.log
```

Od tego momentu należy rozpocząć normalne używanie systemu powstrzymując się od czynności administracyjnych takich jak instalacja i konfiguracja oprogramowania. Jednak jeżeli zajdzie taka potrzeba, wówczas należy przełączyć się w tryb admina poleceniem:

```
gentoo ~ # gradm -a admin
Password:
gentoo ~ #
```

Od tego momentu proces uczenia zostaje wstrzymany. Po zakończeniu czynności administracyjnych należy opuścić tryb admina w celu kontynuowania procesu uczenia.

```
gentoo ~ # gradm -u
gentoo ~ #
```

Warto również zaznaczyć, że logowana jest każda aktywność w systemie plików. Autorzy sugerują powstrzymanie się od wykonywania operacji na dużej ilości plików (np. przeszukiwanie poleceniem find katalogu /). Aby zakończyć proces uczenia można wyłączyć RBAC (*gradm -D*) lub uruchomić ponownie system. Po zakończeniu procesu uczenia w systemie pozostanie plik wskazany w poleceniu rozpoczynającym uczenie, w tym przypadku /etc/grsec/learning.log. Fragment takiego pliku wygląda następująco:

```
default 68    1000    1000    /bin/cat    /    1    1    /lib64/ld-2.22.so    16    0.0.0.0
default 68    1000    1000    /bin/cat    /    1    1    /bin/cat    8    0.0.0.0
default 68    1000    1000    /bin/cat    /    1    1    /lib64/ld-2.22.so    8    0.0.0.0
default 68    1000    1000    /bin/cat    /    1    1    /etc/ld.so.cache    16    0.0.0.0
default 68    1000    1000    /bin/cat    /    1    1    /etc/ld.so.cache    17    0.0.0.0
default 68    1000    1000    /bin/cat    /    1    1    /lib64/libc-2.22.so    16    0.0.0.0
default 68    1000    1000    /bin/cat    /    1    1    /lib64/libc-2.22.so    17    0.0.0.0
default 68    1000    1000    /bin/cat    /    1    1    /lib64/libc-2.22.so    8    0.0.0.0
default 68    1000    1000    /bin/cat    /    1    1    /usr/lib64/locale/locale-
archive    16    0.0.0.0
default 68    1000    1000    /bin/cat    /    1    1    /usr/lib64/locale/locale-
archive    17    0.0.0.0
```

W przedstawionym fragmencie ukazane jest użycie polecenia *cat* i jak widać, zarejestrowane zostały wszystkie pliki z jakich ten program korzysta. Kolejnym etapem w konfiguracji RBAC jest wygenerowanie polityki z zebranego logu. W tym celu należy wykonać polecenie:

```
gentoo ~ # gradm -F -L /etc/grsec/learning.log -O /etc/grsec/learning.roles
Beginning full learning 1st pass...done.
Beginning full learning role reduction...done.
Beginning full learning 2nd pass...done.
Beginning full learning subject reduction for user user...done.
Beginning full learning subject reduction for user root...done.
```

```
Beginning full learning subject reduction for user sshd...done.
Beginning full learning object reduction for subject /...done.
Beginning full learning object reduction for subject /bin/ifconfig...done.
Beginning full learning object reduction for subject /bin/ps...done.
Beginning full learning object reduction for subject /usr/bin/top...done.
Beginning full learning object reduction for subject /...done.
Beginning full learning object reduction for subject /bin/bash...done.
Beginning full learning object reduction for subject /bin/dmesg...done.
Beginning full learning object reduction for subject /bin/login...done.
Beginning full learning object reduction for subject /etc/init.d...done.
Beginning full learning object reduction for subject /sbin/agetty...done.
Beginning full learning object reduction for subject /sbin/init...done.
Beginning full learning object reduction for subject /sbin/openrc...done.
Beginning full learning object reduction for subject /sbin/shutdown...done.
Beginning full learning object reduction for subject /sbin/udevd...done.
Beginning full learning object reduction for subject /usr/sbin/sshd...done.
Full learning complete.
        gentoo ~ #
```

Po przetworzeniu logu widać, że przez czas uczenia zalogowało się trzech użytkowników: root, user i sshd (sshd jest specjalnym użytkownikiem na potrzeby ssh) oraz zostało przeanalizowane każde wykonanie programu. Plikiem wynikowym jest polityka, którą należy dołączyć do standardowej, znajdującej się w pliku */etc/grsec/policy*. Można to wykonać poprzez dołączenie wygenerowanego pliku z polityką do istniejącego lub dopisać (najlepiej na końcu) linię:

```
include </etc/grsec/learning.policy>
```

Można teraz sprawdzić poprawność konfiguracji:

```
        gentoo ~ # gradm -C
Duplicate variable "grsec_denied" defined on line 17 of /etc/grsec/learning.policy.
        gentoo ~ #
```

Pojawił się błąd mówiący o duplikacie zmiennej. Aby rozwiązać ten problem należy zakomentować lub usunąć sekcję redefiniującą tą zmienną (oryginalna zmienna jest w pliku *policy*). Podany numer linii wskazuje koniec sekcji powodującej konflikt. Dodatkowo podobny błąd będzie dotyczył również dwóch ról – *admin* i *shutdown*, które trzeba potraktować analogicznie. W przypadku pliku, który został wygenerowany przez autora, należało usunąć 35 linii. Teraz wystarczy już uruchomić skonfigurowany RBAC:

```
        gentoo ~ # gradm -E
```

Działanie można przetestować np. w taki sposób:

```
        gentoo ~ # cat /etc/grsec/policy
```

```
cat: /etc/grsec/policy: No such file or directory
        gentoo ~ #
```

W logach systemu pojawiła się następująca informacja:

[14417.259336] grsec: From 192.168.0.10: (root:U:/) denied access to hidden file /etc/grsec/policy by /bin/cat[cat:2982] uid/euid:0/0 gid/egid:0/0, parent /bin/bash[bash:2849] uid/euid:0/0 gid/egid:0/0

Czasem jednak proces nauki nie jest dokładnie taki, jakiego by oczekiwał administrator. Skutkuje to potrzebą modyfikacji pliku z polityką. Ogólna postać wpisów jest następująca[18]:

```
role <nazwa roli> <flagi roli>
<dodatkowe atrybuty roli>
subject / <flagi podmiotu> {
        / <uprawnienia obiektu>
        <wyszczególnione ścieżki do plików/katalogów> <uprawnienia obiektu>
        <uprawnienia „Linux Capabilities”>
        <uprawnienia IP>
}
subject <ścieżka do opcjonalnego podmoitu> <flagi podmiotu> {
        / <uprawnienia obiektu>
        <wyszczególnione ścieżki do plików/katalogów> <uprawnienia obiektu>
        …
}
role <nazwa roli 2> <flagi roli>
…
```

Ważne jest by zachować odpowiednią kolejność: definiując daną rolę (słowo kluczowe ***role***), wszystkie linie po jej definicji odnoszą się do niej aż do końca pliku z polityką lub kolejnej definicji roli. Analogicznie wygląda sytuacja z z podmiotami (słowo kluczowe ***subject***). Minimalna definicja roli musi zawierać definicję roli oraz podmiot „ / ”. Na szablonie zostało to oznaczone poprzez podkreślenie. Poniżej przykładowa minimalna definicja roli, która pozwala na wszystko:

```
role admin sA
subject / rakv {
        /rwxcdmli
}
```

Poniżej zostaną przedstawione najczęściej używane flagi i uprawnienia:

Nazwa roli

Nazwą roli może być istniejący użytkownik w systemie, istniejąca grupa, słowo kluczowe *default* lub ciąg znaków nie będących nazwą użytkownika w systemie bądź nazwą grupy

[18] https://en.wikibooks.org/wiki/Grsecurity/The_RBAC_System - data odczytu 9.01.2017 r.

w systemie – wówczas będzie to definicja roli specjalnej (więcej informacji poniżej). Podczas procesu logowania, sprawdzane jest czy istnieje rola o takiej samej nazwie jak login, jeżeli istnieje, użytkownikowi zostaje przydzielona odpowiednia rola. W przypadku braku zgodności następuje próba dopasowania grupy logującego się użytkownika do listy ról. Jeżeli zostanie znaleziona taka rola, to użytkownikowi zostaje przydzielona rola grupy. W przeciwnym przypadku użytkownikowi zostaje przypisana rola *default*. Należy również zwrócić uwagę na pewien fakt – jeżeli rola użytkownika będzie miała wpis zabraniający używania takiej roli zdalnie (zwykle taki wpis jest tworzony na początku uczenia się systemu), to użytkownikowi będzie przypisana rola *default*!

Flagi roli

- ***u*** – rola użytkownika. Nazwą roli musi być login istniejącego użytkownika w systemie. Stosuje się tą opcję do przydzielania uprawnień konkretnemu użytkownikowi.
- ***g*** – rola grupy. Nazwą roli musi być istniejąca grupa w systemie. Zastosowanie podobne jak roli użytkownika, tylko wpływa na wszystkich użytkowników w danej grupie.
- ***s*** – rola specjalna. Rola występująca wyłącznie w RBAC, do której można się przełączyć przez polecenie *gradm -p <nazwa roli>*. Zwykle stosowane w celu podniesienia uprawnień.
- ***l*** – rola z włączoną możliwością nauki. Podczas procedury nauki, tylko dla tych ról, które zostały oznaczonych tą flagą zostaną zebrane logi oraz zostanie wygenerowana polityka.
- ***A*** – rola administracyjna. Dla takich ról zostają zniesione restrykcje dotyczące ładowań bibliotek oraz mechanizmu *ptrace*.
- ***G*** – umożliwienie roli używanie polecenia *gradm.*
- ***N*** – wyłączenie mechanizmu uwierzytelniania przy przełączaniu na tą rolę. Przełączanie poprzez polecenie *gradm -n <nazwa roli>*
- ***P*** – do uwierzytelniania używany jest podsystem PAM.

Dodatkowe atrybuty roli

- ***role_transitions*** – umożliwienie roli przełączanie się na inne role specjalne, które zostały wymienione. Przykład: *role_transitions developers staff*
- ***role_allow_ip*** – umożliwienie uwierzytelnianie do roli z podanego adresu/podsieci podanego w formacie CIDR. Podanie adresu 0.0.0.0/32 uniemożliwia zdalne uwierzytelnianie. Przykład: *rule_allow_ip 192.168.1.0/24*

Flagi podmiotu

- ***a*** **–** umożliwienie procesowi dostępu do /dev/grsec
- ***h*** **–** ukrycie procesu.
- ***v*** – proces może widzieć ukryte procesy
- ***p*** **–** proces jest chroniony przed zabiciem. Jedynie procesy z tego samego podmiotu oraz te z flagą *k* mogą go zabić.
- ***k*** **–** proces może zabić procesy ukryte
- ***r*** **–** znieś ograniczenia związane z użyciem *ptrace*
- ***o*** – nadpisz dziedziczone uprawnienia

Uprawnienia obiektu

- ***r*** **–** obiekt może być otwarty w trybie do odczytu
- ***w*** **–** obiekt może być otwarty w trybie do zapisu i do dopisywania (appending)
- ***a*** **–** obiekt może być otwarty w trybie do dopisywania (appending), co skutkuje tym, że nie można w nim zmienić już istniejących danych (przykład zastosowania: logi)
- ***x*** **–** obiekt może zostać uruchomiony
- ***c*** **–** umożliwienie tworzenia plików/katalogów
- ***d*** – umożliwienie usuwania plików/katalogów
- ***h*** **–** obiekt jest ukryty
- ***m*** **–** obiektowi można nadać bity setuig/setgid
- ***i*** **–** wykonaj obiekt z uprawnieniami odziedziczonymi po podmiocie, z którego został wykonany. Dotyczy tylko plików wykonywalnych
- ***l*** **–** umożliwia tworzenie dowiązań twardych (hardlink)
- \- (puste miejsce) wpisanie ścieżki do obiektu bez uprawnień umożliwia tylko odczyt metadanych obiektu (m.in. rozmiaru, daty utworzenia, uprawnień, nazwy). Nie daje to jakiegokolwiek dostępu do zawartości obiektu.

Uprawnienia „Linux Capabilities”

Dostęp do Capabilites może być przyznawany lub blokowany poprzez dopisanie „+” lub „-” przed nazwą danego Capability. Grsecurity wprowadza dodatkową pozycję w liście dostępnych Capabilities – *CAP_ALL* – które jest uwzględnieniem wszystkich Capabilites.

Przykładowo, jeżeli istnieje potrzeba wyłączenia wszystkich Capabilites poza możliwością zmiany właściciela pliku/katalogu, w wyznaczonym miejscu należy dopisać:

-CAP_ALL
+CAP_CHOWN

Lista wszystkich Capabilites jest dostępna w podręczniku systemowym (*man capabilites*) lub tu: https://en.wikibooks.org/wiki/Grsecurity/Appendix/Capability_Names_and_Descriptions.

Uprawnienia IP

Uprawnienia związane z połączeniami sieciowymi są podzielone na dwa typy: połączenia wychodzące (słowo kluczowe ***connect***) i połączenia przychodzące (słowo kluczowe ***bind***). Szablon takich definicji wygląda w następujący sposób:

```
connect/bind <!> <IP/nazwa hosta>/<maska>:<port/zakres portów> <typy gniazd sieciowych> <protokoły>
```

lub

```
connect/bind disabled
```

Typy gniazd sieciowych są następujące: *ip, dgram, stream, raw_sock, rdm, any_sock*. Protokoły są takie jak w pliku /etc/protocols lub można użyć słowa kluczowego *any_proto*. Wymaganymi polami jest tylko *<IP/nazwa hosta>*, natomiast jeżeli nie jest konieczny adres, wówczas należy wpisać 0.0.0.0. Można jeszcze w to pole wpisać nazwę interfejsu sieciowego. Dopuszczalne jest również wyszczególnianie vlanu w formie *<nazwa interfejsu>#<numer vlanu>*. Wykrzyknik jest używany do negacji. Przykładowe wpisy:

```
bind eth0#2:22 stream tcp
```

Podmiot może nasłuchiwać na vlanie numer 2 na interfejsie eth0 na portcie 22 i może przyjmować połączenia tcp.

```
connect 0.0.0.0/0 1234-4321 dgram udp
```

Podmiot może wysyłać pakiety udp do dowolnego hosta na portach docelowych od 1234 do 4321.

```
bind ! 12.34.0.0/16:80 stream tcp
```

Podmiot nie może przyjmować połączeń tcp z podsieci 12.34.0.0/16 na porcie 80.

Poniżej przedstawiona jest przykładowa definicja roli użytkownika *user* z nauczonym zachowaniem dla programu *dmesg*:

```
role user uG
role_transitions admin shutdown
role_allow_ip   192.168.0.0/16
role_allow_ip   172.16.0.0/12
role_allow_ip   10.0.0.0/8
subject / {
      /             r
```

```
        /opt            rx
        /home/user      rwxcd
        /mnt            rw
        /dev
        /dev/urandom    r
        /dev/random     r
        /dev/zero       rw
        /dev/input      rw
        /dev/psaux      rw
        /dev/null       rw
        /dev/tty?       Rw
        /dev/console    rw
        /dev/tty        rw
        /dev/pts        rw
        /dev/ptmx       rw
        /dev/mixer      rw
        /dev/cdrom      r
        /bin            rx
        /sbin           rx
        /lib            rx
        /lib32          rx
        /lib64          rx
        /usr            rx
        /usr/src        h
        /etc            rx
        /proc           rwx
        /proc/sys       r
        /sys            h
        /root           h
        /run            r
        /tmp            rwcd
        /var            rwxcd
        /var/tmp        rwcd
        /var/log

        -CAP_ALL
        +CAP_KILL
        +CAP_NET_RAW
        +CAP_NET_ADMIN
        +CAP_NET_RAW

        bind disabled
}
subject /bin/dmesg o {
        /                       h
        /bin                    h
        /bin/dmesg                  x
        /dev                    h
        /dev/kmsg                   r
        /etc                    h
```

```
    /etc/ld.so.cache                r
    /lib64                          h
    /lib64/ld-2.22.so               x
    /lib64/libc-2.22.so             rx
    /usr                            h
    /usr/lib64/gconv/gconv-modules.cache    r
    /usr/lib64/locale/locale-archive        r
    -CAP_ALL
    +CAP_SYSLOG
    bind    disabled
    connect disabled
}
```

Ostatecznie najlepszy rezultat osiągnie się wtedy, kiedy połączy się ręczną edycję wraz z wynikami procesu nauki.

3. Podsumowanie

W artykule przedstawiono źródła zagrożeń systemu, w tym między innymi polegające na uzyskaniu przez nieuprawniony podmiot dostępu do niego. Taki dostęp może być powodem utraty danych czy tylko reputacji, a zatem może skończyć się bezpośrednimi stratami ekonomicznymi lub tylko prestiżu, co w konsekwencji może również doprowadzić do strat finansowych.

Autorzy w artykule wskazali kilka podstawowych, najczęściej występujących zagrożeń. Nie poruszając tematyki zabezpieczenia przed nimi, przedstawli sposoby ograniczenia skutków po udanej próbie włamania. W przypadku takiego udanego włamania podsystemy obowiązkowej kontroli systemu (MAC) w postaci *SELinux* (domyślnie aktywny w takich systemach jak CentOS, Fedora czy RedHat Enterprise Linux) lub *RBAC* z łatką *grsecurity* mogą uniemożliwić dostęp włamywaczowi do zasobów, do których aplikacja nie musiała mieć dostępu aby poprawnie działać.

3.1. Spis literatury

1. Foster J. C., *Buffer Overflow Attacks: Detect, Exploit, Prevent,* Syngress Publishing, Rockland 2005
2. Frisch E., *Unix – Administracja system – drugie wydanie,* Oficyna Wydawncza READ ME, Łódź 1997
3. McConnell S., *Code complete, 2nd edition*, Microsoft Press, Redmond 2004
4. Silberschatz A., Galvin P.B., *Podstawy systemów operacyjnych*, Wydawnictwo Naukowo-Techniczne, Warszawa 2002
5. Vermeulen S., *SELinux System Administration*, Packt Publishing 2013

3.2. Netografia

1) https://www.theguardian.com/technology/2015/aug/10/carphone-warehouse-uk-data-watchdog-investigating-customer-hack - data odczytu, 6.11.2016 r.
2) https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40 - data odczytu, 6.11.2016 r.
3) http://vms.drweb-av.pl/virus/?_is=1&i=4323517 – data odczytu, 6.11.2016 r.

4) https://www.aldeid.com/wiki/Exploits/proftpd-1.3.3c-backdoor – data odczytu 6.11.2016 r.
5) https://access.redhat.com/blogs/766093/posts/1975883 – data odczytu 6.11.2016 r.
6) https://www.openhub.net/p/apache - data odczytu 6.11.2016 r.
7) https://wiki.gentoo.org/wiki/SELinux/Tutorials/How_does_a_process_get_into_a_certain_context - data odczytu 18.12.2016 r.
8) J. McDonald, *Defeating Solaris/SPARC non-executable stack protection*. Bugtraq, Mar. 1999, Online: http://seclists.org/bugtraq/1999/Mar/4.
9) M. Ivaldi, *Re: Older SPARC return-into-libc exploits. Penetration Testing*, Aug. 2007, Online:http://seclists.org/pen-test/2007/Aug/214
10) https://en.wikibooks.org/wiki/Grsecurity/The_RBAC_System

Streszczenie

Niniejsza artykuł ma na celu zapoznanie czytelnika z najpopularniejszymi lukami w zabezpieczeniach systemu operacyjnego Linux oraz propozycjami zabezpieczenia się po pozytywnym skorzystaniu z nich przez intruza w celu zabezpieczenia systemu komputerowego przed sięgnięciem do zasobów innych niż skompromitowana usługa.