PROGRAMOWANIE DLA PROFESJONALISTÓW BEZPIECZEŃSTWA

Jako profesjonalista ds. bezpieczeństwa musisz wiedzieć, jak hakerzy i testerzy bezpieczeństwa wykorzystują programowanie komputerowe. Ten moduł opisuje podstawowe umiejętności programowania. Po ukończeniu tego modułu nie zostaniesz ekspertem programistą, ale będziesz mieć jaśniejsze pojęcie o tym, jak pisane są programy. Usunięcie tej tajemnicy eliminuje strach, którego wielu profesjonalistów sieciowych doświadcza, słysząc słowo „programowanie". Podstawowa znajomość programowania może również pomóc w opracowywaniu niestandardowych narzędzi bezpieczeństwa lub modyfikowaniu istniejących narzędzi podczas przeprowadzania testów bezpieczeństwa. W rzeczywistości większość opisów stanowisk dla profesjonalnych testerów bezpieczeństwa obejmuje wymóg tworzenia niestandardowych narzędzi bezpieczeństwa. Tak jak dobry cieśla wie, jak zmodyfikować narzędzie, aby pasowało do specjalnego zadania, testerzy bezpieczeństwa powinni wiedzieć, jak zmodyfikować narzędzia komputerowe stworzone w jednym celu, aby można je było wykorzystać do innych funkcji. Ten moduł daje ogólny przegląd języków C, HTML, Perl i Python. Zostanie programistą wymaga dużo czasu i praktyki, ale ten moduł daje możliwość zbadania niektórych programów i ćwiczenia samodzielnego pisania kilku z nich.

WPROWADZENIE DO PROGRAMOWANIA KOMPUTEROWEGO

Podobnie jak redaktorzy książek muszą rozumieć zasady i składnię języka angielskiego, programiści komputerowi muszą rozumieć zasady języków programowania i radzić sobie z błędami składniowymi. Składnia polecenia musi być dokładna, aż po umieszczenie średników i nawiasów. Jeden niewielki błąd i program nie będzie działał poprawnie, a co gorsza, przyniesie nieprzewidywalne rezultaty. Bycie programistą wymaga bystrego oka i cierpliwości; pamiętaj, że błędy nie są niczym niezwykłym, gdy po raz pierwszy próbujesz stworzyć program. Niestety, większość uczelni nie uczy programowania z myślą o bezpieczeństwie. Wiele obecnych ataków na systemy operacyjne i aplikacje jest możliwych z powodu złych praktyk programistycznych. Mary Ann Davidson, dyrektor ds. bezpieczeństwa (CSO) Oracle, przemawia na ten temat na całym świecie. Twierdzi, że programiści oprogramowania skupiają się na „fajnej technologii" i najnowszych językach programowania. „Oni nie myślą jak atakujący" — powiedziała przed publicznością składającą się z ponad 1000 profesjonalistów ds. zapewniania bezpieczeństwa informacji. „Nie ma też wymogu, aby programiści oprogramowania wykazali się biegłością w bezpiecznym, zabezpieczonym programowaniu jako warunkiem przyjęcia do szkoły" — dodała. Szczegóły na ten temat wykraczają poza zakres tej książki, ale jeśli zdecydujesz się na programowanie lub inżynierię oprogramowania jako kierunek studiów, nalegaj na uczelnię, do której się wybierasz, aby uwzględniła bezpieczne programowanie. CSO firmy Oracle przedstawiła kilka sugestii dotyczących zmiany systemu szkolnictwa wyższego. Uważa, że bezpieczeństwo powinno być częścią każdego kursu informatyki, „a nie tylko pojedynczego kursu, który studenci archiwizują i zapominają", a podręczniki do informatyki powinny być pisane z naciskiem na bezpieczne techniki programowania. Oceny powinny być częściowo oparte na „hackowalności" kodu przesyłanego przez studentów do zadań, a studenci powinni być zobowiązani do korzystania z automatycznych narzędzi w celu znalezienia luk w swoim kodzie. Bezpieczeństwo musi być zintegrowane z każdym projektem inżynierii oprogramowania od samego początku, a nie po fakcie. Nowa dziedzina cyberbezpieczeństwa o nazwie SecDevOps próbuje zmienić sposób myślenia programistów, aby rozwijali kod z myślą o bezpieczeństwie i operacjach informatycznych. Termin SecDevOps oznacza bezpieczeństwo, rozwój i operacje. Tworzenie kodu z myślą o bezpieczeństwie i zrozumienie operacji IT skutkuje bardziej solidnym i odpornym na ataki oprogramowaniem. Zamiast dodawać zabezpieczenia poprzez zmianę czyjegoś kodu, aby uczynić go bardziej bezpiecznym, SecDevOps próbuje sprawić, aby pierwotni programiści „wpiekli" zabezpieczenia w kod od samego początku. Ten moduł ma na celu pobudzić Twój apetyt i dać Ci przegląd programowania. Na początek zapoznaj się z podstawami programowania.

Podstawy programowania

Podręczniki wypełnione składnią i poleceniami języka programowania mogą zająć dużo miejsca na półkach, ale możesz nauczyć się podstaw dowolnego języka programowania bez korzystania z podręczników lub przewodników online. W rzeczywistości możesz zacząć pisać programy, mając jedynie pewną wiedzę na temat podstaw programowania, które możesz zapamiętać dzięki akronimowi BLT (jak w bekonie, sałata i pomidor): rozgałęzianie, pętlenie i testowanie.

Rozgałęzianie, pętlenie i testowanie (BLT)

Większość języków programowania ma sposób na rozgałęzianie, pętlenie i testowanie. Na przykład funkcja w programie C może rozgałęziać się do innej funkcji w programie, wykonywać tam zadanie, a następnie wracać do punktu początkowego. Funkcja to mini program w programie głównym, który wykonuje zadanie. Na przykład możesz napisać funkcję, która dodaje dwie liczby, a następnie zwraca odpowiedź do funkcji, która ją wywołała. Rozgałęzianie przenosi Cię z jednego obszaru programu (funkcji) do innego obszaru. Pętla to wykonywanie zadania w kółko. Pętla zwykle kończy się po przeprowadzeniu testu zmiennej i zwróceniu wartości true lub false. Chociaż na razie nie musisz martwić się składnią, sprawdź poniższy program, aby zobaczyć, gdzie używa rozgałęzień, pętli i testowania:

```
#include <stdio.h>

main()

{

int a = 1; // Variable initialized as integer, value 1

if (a > 2) ; // Testing whether "a" is greater than 2

printf ("a is greater than 2");

else

GetOut(); // Branching: calling a different function

GetOut() // Do something interesting here

{

for (int a=1; a<11; a + + ) // Loop to display 10 times

{

printf("I'm in the GetOut() function") ;

}

}

}
```

Oto ona: BLT programowania komputerowego. W programowaniu jest wiele do nauczenia, ale wiedząc, jak wykonać te trzy czynności, możesz zbadać program i zrozumieć jego funkcjonalność. Program zawiera różne funkcje lub moduły, które wykonują określone zadania. Powiedzmy, że piszesz program do robienia kanapki BLT. Pierwszym krokiem jest wypisanie zadań w tym procesie. W żargonie komputerowym piszesz algorytm (przepis) do robienia kanapki BLT. Starasz się, aby algorytm był tak

prosty, jak to możliwe, ale tworzenie algorytmu jest jedną z najważniejszych umiejętności programistycznych do opanowania. Pominięcie kroku w algorytmie może powodować problemy. Na przykład nieopłukanie sałaty może skutkować błędem w kanapce. Podobnie, nieuważne przejrzenie kodu programu może skutkować błędem w programie — błędem powodującym nieprzewidywalne rezultaty. Błędy są gorsze niż błędy składniowe, ponieważ program może działać pomyślnie z błędem, ale dane wyjściowe mogą być nieprawidłowe lub niespójne. Wykonywanie zadań w nieprawidłowej kolejności może również powodować zamieszanie. Na przykład, posmarowanie chleba majonezem przed zapiekaniem może spowodować, że będzie rozmiękły. Poniższa lista to przykład algorytmu robienia kanapki BLT:

• Kup składniki.

• Zbierz wszystkie przybory potrzebne do zrobienia kanapki.

• Umyj pomidory i sałatę.

• Pokrój pomidory w plastry i rozdziel liście sałaty.

• Usmaż bekon.

• Odcedź bekon.

• Zapiecz chleb.

• Nałóż majonez na tosty.

• Na tosty połóż smażony bekon, pokrojonego pomidora i liście sałaty.

• Połącz dwie kromki zapieczonego chleba.

Następnie programista zamieniłby ten algorytm na pseudokod. Pseudokod nie jest językiem programowania; to język podobny do angielskiego, którego możesz użyć, aby pomóc w tworzeniu struktury swojego programu. Poniższy przykład to pseudokod, który dotyczy zakupu wszystkich składników potrzebnych do kanapki BLT przed napisaniem kodu programowania:

PurchaseIngredients Function

Call GetCar Function

Call DriveToStore Function

Purchase Bacon, Bread, Tomatoes, Lettuce, and Mayonnaise at store

End PurchaseIngredients Function

Po napisaniu pseudokodu możesz zacząć pisać swój program w wybranym przez siebie języku. Czy szkicowanie algorytmu i pisanie pseudokodu jest konieczne dla każdego programu komputerowego, który piszesz? Nie. Jeśli program, który piszesz, ma bardzo mało linii kodu, możesz pominąć te kroki, ale dla początkujących programistów te dwa kroki są pomocne

Dokumentacja

Podczas pisania dowolnego programu musisz udokumentować swoją pracę. Aby to zrobić, dodajesz komentarze do kodu, które wyjaśniają, co robisz. Dokumentacja nie tylko ułatwia komuś modyfikowanie programu; pomaga również zapamiętać, co myślałeś, pisząc program. Fraza „Bez komentarza” może być odpowiednia dla polityków lub inwestorów z Wall Street, którzy mają dostęp

do informacji poufnych, ale nie dla programistów. Chociaż dokumentacja jest ważna, wielu programistów uważa ją za czasochłonną i żmudną. Często uważają, że ich kod jest oczywisty i wystarczająco łatwy do utrzymania i modyfikowania przez każdego, więc dokumentowanie ich pracy nie jest konieczne. Szybko jednak odkryjesz, że bez dobrej dokumentacji nie zrozumiesz wierszy kodu, które napisałeś trzy tygodnie temu, a co dopiero oczekiwać, że obcy zrozumie Twój tok myślenia. Na przykład następujące komentarze mogą pomóc następnemu programiście zrozumieć, dlaczego do istniejącego programu dodano nową funkcję:

/* Następująca funkcja została dodana do programu 15 czerwca 2021 r. na prośbę Działu Marketingu. Raporty generowane przez funkcję sales() nie dostarczały pracownikom marketingu informacji o sprzedaży w Azji. Ta nowa funkcja wykorzystuje dane z plików tekstowych z biur w Tokio i Hongkongu. - Kendra Choi */

Firmy zajmujące się inżynierią oprogramowania nie zatrudniają programistów, którzy nie dokumentują swojej pracy, ponieważ wiedzą, że 80 procent kosztów projektów oprogramowania to konserwacja. Wiedzą również, że średnio 10 błędów na każde 1000 wierszy kodu to standard branżowy. Na przykład szacuje się, że system Windows 10 zawiera ponad 50 milionów wierszy kodu, ale inżynierowie oprogramowania firmy Microsoft, częściowo ze względu na ścisłe zasady dokumentacji i praktyki Secure Software Development Lifecycle Practices, mogą ograniczyć liczbę błędów do mniejszej niż średnia. Ogólnie rzecz biorąc, średnia liczba błędów w kodzie firmy Microsoft jest poniżej standardu branżowego. Jednak biorąc pod uwagę, że błędy są tak powszechne w wielu programach, łatwo zrozumieć, w jaki sposób atakujący mogą odkrywać luki w zabezpieczeniach oprogramowania. Programiści mogą łatwo przeoczyć problemy w tysiącach wierszy kodu, które mogą stworzyć lukę w zabezpieczeniach, którą atakujący mogą wykorzystać.

BAJTY BEZPIECZEŃSTWA

W lipcu 2021 r. firma Microsoft wydała awaryjną poprawkę na krytyczny błąd nazwany PrintNightmare. Błąd został przypadkowo ujawniony przez badaczy. Korzystając z tej luki, hakerzy mogli instalować programy, przeglądać, zmieniać i usuwać dane lub tworzyć nowe konta z pełnymi uprawnieniami użytkownika zdalnie we wszystkich wersjach systemu Windows. Błąd dotyczył luki w zabezpieczeniach usługi Windows Print Spooler, części systemu Windows zarządzającej drukowaniem. System Windows 7 osiągnął koniec swojego cyklu życia i nie jest już aktywnie wspierany przez firmę Microsoft, ale błąd był na tyle poważny, że firma Microsoft wydała również poprawkę dla systemu Windows 7.

Napisanie pierwszego algorytmu

Czas trwania: 10 minut

Cel: Naucz się pisać algorytm.

Opis: Programiści muszą podchodzić do rozwiązywania problemów w logicznych krokach lub zadaniach. Pominięcie kroku może mieć katastrofalne skutki, dlatego powinieneś nauczyć się myśleć w sposób ustrukturyzowany i logiczny. Dobrym sposobem na sprawdzenie, czy potrafisz postępować zgodnie z podejściem krok po kroku, jest wykonywanie ćwiczeń, które zachęcają Cię do myślenia w ten sposób. W przypadku tego ćwiczenia wypisz co najmniej siedem kroków, aby zrobić jajecznicę. Podczas pisania kroków upewnij się, że niczego nie bierzesz za pewnik. Załóż, że ktoś, kto nie ma pojęcia o gotowaniu — ani nawet o jajkach — spróbuje postępować zgodnie z Twoim algorytmem.

NAUKA JĘZYKA C

Testerzy ds. bezpieczeństwa mają do dyspozycji wiele języków programowania. Rozpoczniesz swoją podróż od wprowadzenia do jednego z najpopularniejszych języków programowania: C, opracowanego przez Dennisa Ritchiego w Bell Laboratories w 1972 roku. Język C jest zarówno potężny, jak i zwięzły. W rzeczywistości UNIX, który został pierwotnie napisany w języku asemblera, wkrótce został przepisany w języku C. Niewielu programistów chce pisać programy w kodzie binarnym (kod maszynowy) lub języku maszynowym, dlatego opracowano język asemblera. Używa on kombinacji liczb szesnastkowych i wyrażeń, takich jak mov, add i sub, więc pisanie programów w tym języku jest łatwiejsze niż w języku maszynowym. Ten moduł daje podstawowy przegląd języka C. Na wielu uczelniach cały kurs poświęcony jest nauce tego języka; inne pomijają C i uczą C++, rozszerzenia języka C. Wielu profesjonalistów ds. bezpieczeństwa i hakerów nadal używa C ze względu na jego moc i użyteczność na wielu platformach. Jednak specjaliści ds. bezpieczeństwa uważają również, że C jest jednym z najmniej bezpiecznych języków programowania ze względu na podatność na przepełnienia bufora. Dlatego też znajomość języka jest pomocna, aby uniknąć jego pułapek. Kompilator to program, który konwertuje program tekstowy, zwany kodem źródłowym, na kod wykonywalny lub binarny. Tabela zawiera listę dostępnych kompilatorów C.

Kompilator: Opis

Kompilatory Intel dla systemów Windows i Linux: Kompilator Intel C++ jest przeznaczony do tworzenia aplikacji dla serwerów Windows, komputerów stacjonarnych, laptopów i urządzeń mobilnych. Kompilator Intel Linux C++ twierdzi, że optymalizuje szybkość dostępu do informacji z bazy danych MySQL, programu bazy danych typu open source używanego przez wiele korporacji i firm e-commerce.

Microsoft Visual C++: Kompilator Ten kompilator jest szeroko używany przez programistów tworzących aplikacje C i C++ dla platform Windows.

Kompilatory GNU C i C++ (GCC): Te bezpłatne kompilatory można pobrać dla platform Windows i *nix. Większość systemów *nix zawiera kompilator GNU GCC

Większość kompilatorów C może również tworzyć programy wykonywalne w C++. Kompilatory Intel i Microsoft muszą zostać zakupione, ale wiele innych kompilatorów jest bezpłatnych i można je znaleźć, wyszukując w Internecie. Kompilatory online nie wymagają, aby programista miał lokalnie zainstalowany kompilator na swoim komputerze. Te kompilatory online mają strony internetowe, które akceptują kod od programisty, a następnie kompilują go na serwerach dostawcy kompilatora online, używając odpowiedniego kompilatora dla używanego języka programowania. Kompilatory online są dobre do eksperymentów i nauki, ale nie są praktyczne w programowaniu w świecie rzeczywistym. Kwestie bezpieczeństwa związane z przesyłaniem kodu na serwer zewnętrzny i powolność kompilacji dużych projektów sprawiają, że kompilatory online są nieodpowiednie i niepraktyczne w przypadku poważnego programowania.

UWAGA

Niebezpieczne w C jest to, że początkujący może popełnić kilka dużych błędów. Na przykład programista może przypadkowo zapisać dane w obszarach pamięci, co może spowodować awarię programu lub, co gorsza, dać atakującemu możliwość przejęcia kontroli nad zdalnym systemem. Zazwyczaj zapisuje się kod wykonywalny, który może dać atakującemu tylne wejście do systemu, rozszerzyć uprawnienia atakującego do uprawnień administratora lub po prostu spowodować awarię programu. Ten typ ataku jest zwykle możliwy, ponieważ programista nie sprawdził danych wprowadzanych przez użytkowników. Na przykład, jeśli użytkownicy mogą wprowadzić 300 znaków, gdy zostaną poproszeni o podanie nazwiska, atakujący prawdopodobnie może wprowadzić kod wykonywalny w tym momencie programu. Kiedy widzisz termin „podatność na przepełnienie bufora”,

pomyśl „złe praktyki programistyczne”. Pamiętaj, że chociaż C jest łatwy do nauczenia i używania, błędy w jego używaniu mogą spowodować uszkodzenie systemu.

Anatomia programu C

Wielu doświadczonych programistów nie potrafi myśleć o języku C bez pamiętania programu „Hello, world!”, pierwszego programu, którego uczy się student C:

```
/* The famous "Hello, world!" C program */

#include <stdio.h> /* Load the standard IO library. The library contains functions your C

program might need to call to perform various tasks. */

main()

{

printf("Hello, world!\n\n") ;

}
```

To wszystko. Możesz napisać te wiersze kodu w prawie każdym edytorze tekstu, takim jak Notatnik, jeśli używasz systemu Windows, lub edytor vim, jeśli używasz systemu Linux. Jeśli chcesz mieć więcej funkcji edycji niż w Notatniku lub vim, możesz zainstalować i używać Notepad++ w systemie Windows lub użyć gedit w systemie Linux. Możesz pobrać Notepad++ ze strony https://notepad-plus-plus.org/downloads/. Poniższe sekcje wyjaśniają każdy wiersz kodu w tym programie. Wiele programów C używa symboli /* i */ do ujmowania długich komentarzy zamiast używania symboli // do komentarzy jednowierszowych. Na przykład możesz wpisać symbole /*, dodać tyle wierszy tekstu komentarza, ile potrzeba, a następnie wpisać symbole zamykające */. Zapomnienie o dodaniu */ na końcu tekstu komentarza może spowodować błędy podczas kompilacji programu, więc bądź ostrożny. Instrukcja #include służy do ładowania bibliotek, które zawierają polecenia i funkcje używane w programie. W Hello, world! na przykład polecenie #include <stdio.h> ładuje bibliotekę stdio.h, która zawiera wiele funkcji C. Nawiasy w C oznaczają, że masz do czynienia z funkcją. Programy C muszą zawierać funkcję main(), ale możesz również dodawać własne funkcje do programu C. Zwróć uwagę, że po funkcji main() nawias otwierający (symbol {) znajduje się w osobnym wierszu. Nawiasy pokazują, gdzie zaczyna się i kończy blok kodu. W programie Hello, world! nawias zamykający wskazuje koniec programu. Zapomnienie o dodaniu nawiasu zamykającego jest częstym błędem. Wewnątrz funkcji main() program wywołuje inną funkcję: printf(). Kiedy funkcja wywołuje inną funkcję, używa parametrów, znanych również jako argumenty. Parametry są umieszczane między nawiasami otwierającymi i zamykającymi. W tym przykładzie parametry „Hello, world! \n\n” są przekazywane do funkcji printf(). Funkcja printf() wyświetla następnie (drukuje) słowa „Hello, world!” na ekranie, a znaki \n\n dodają dwie nowe linie po wyświetleniu Hello, world!. Tabela zawiera listę niektórych znaków specjalnych, których można używać z funkcją printf().

Znak: Opis

\n: Nowa linia

\t: Karta

Deklarowanie zmiennych

Zmienna reprezentuje wartość liczbową lub ciąg znaków. Na przykład możesz rozwiązać równanie x + y = z, jeśli znasz dwie wartości zmiennych. W programowaniu możesz deklarować zmienne na początku

programu, aby obliczenia mogły być wykonywane bez interwencji użytkownika. Zmienna może być zdefiniowana jako znak lub znaki, takie jak litery alfabetu, lub może być jej przypisana wartość liczbowa, jak w wyrażeniu int x = 1. Tabela 7-3 przedstawia niektóre typy zmiennych używane w języku C.

Typ zmiennej : Opis

Int: Użyj tego typu zmiennej dla liczby całkowitej (dodatniej lub ujemnej).

Float: Ten typ zmiennej jest dla liczby rzeczywistej zawierającej przecinek dziesiętny, takiej jak 1,299999.

Double: Użyj tego typu zmiennej dla liczby zmiennoprzecinkowej podwójnej precyzji.

Char: Ten typ zmiennej przechowuje wartość pojedynczej litery.

String: Ten typ zmiennej przechowuje wartość wielu znaków lub słów.

Const: Zmienna stała jest tworzona w celu przechowywania wartości, która nie zmienia się przez cały czas trwania programu. Na przykład możesz utworzyć zmienną stałą o nazwie TAX i nadać jej określoną wartość: const TAX =.085. Jeśli ta zmienna jest używana w obszarach programu, które obliczają całkowite koszty po dodaniu podatku w wysokości 8,5%, łatwiej jest zmienić wartość stałej na inną liczbę, jeśli stawka podatku ulegnie zmianie, zamiast zmieniać każde wystąpienie 8,5% na 8,6%.

Jeśli funkcja printf() zawiera wartości inne niż zdanie w cudzysłowie, takie jak liczby, należy użyć specyfikatorów konwersji. Specyfikator konwersji informuje kompilator, jak przekonwertować wartość w funkcji. Na przykład printf("Twoje imię to %s! ", name) ; wyświetla następujące informacje, jeśli zmiennej łańcuchowej o nazwie name przypisano wartość Sue:

Twoje imię to Sue!

Tabela zawiera specyfikatory konwersji dla funkcji printf().

Specyfikator: Typ

%c: Znak

%d: Liczba dziesiętna

%f: Liczba dziesiętna zmiennoprzecinkowa lub liczba podwójna

%s: Ciąg znaków

Oprócz specyfikatorów konwersji programiści używają operatorów do porównywania wartości, wykonywania obliczeń matematycznych i tym podobnych. Najprawdopodobniej programy, które piszesz, będą wymagały obliczania wartości na podstawie operacji matematycznych, takich jak dodawanie lub odejmowanie. Tabela opisuje operatory matematyczne używane w języku C.

Operator : Opis

+ (jednoargumentowy): Nie zmienia wartości liczby. Operatorzy jednoargumentowi używają jednego argumentu; operatorzy binarni używają dwóch argumentów. Przykład: +(2).

- (jednoargumentowy): Zwraca ujemną wartość pojedynczej liczby.

++ (jednoargumentowy): Zwiększa wartość jednoargumentową o 1. Na przykład, jeśli a jest równe 5, ++a zmienia wartość na 6.

- (jednoargumentowy): Zmniejsza wartość jednoargumentową o 1. Na przykład, jeśli a jest równe 5, -a zmienia wartość na 4.

+ (binarny): Dodawanie. Na przykład, a + b.

- (binarny): Odejmowanie. Na przykład, a – b.

* (binarny): Mnożenie. Na przykład, a * b.

/ (binarny): Dzielenie. Na przykład, a / b.

% (binarny): Moduł. Na przykład, 10 % 3 jest równe 1, ponieważ 10 podzielone przez 3 daje resztę równą 1.

Podczas pisania programu w języku C może być również konieczne sprawdzenie, czy warunek jest prawdziwy czy fałszywy. Aby to zrobić, musisz zrozumieć, jak używać operatorów relacyjnych i logicznych, opisanych w tabeli 7-6.

Operator :Opis

== : Operator równości; porównuje równość dwóch zmiennych. Na przykład w a == b warunek jest prawdziwy, jeśli zmienna a jest równa zmiennej b.

! = : Nierówne; wykrzyknik neguje znak równości. Na przykład stwierdzenie if a != b jest czytane jako „jeśli a nie jest równe b”.

> : Większe niż.

< : Mniejsze niż.

>= : Większe lub równe.

<= : Mniejsze lub równe.

&& : Operator AND; jest oceniany jako prawdziwy, jeśli obie strony operatora są prawdziwe. Na przykład if ((a > 5) && : (b > 5)) printf ("Hello, world!"); drukuje tylko wtedy, gdy zarówno a, jak i b są większe niż 5.

| | : Operator OR; jest oceniany jako prawdziwy, jeśli którakolwiek strona operatora jest prawdziwa.

! : Operator NOT; stwierdzenie ! Na przykład (a == b) jest oceniane jako prawda, jeśli a nie jest równe b.

Używając operatorów przypisania złożonego jako metody skróconej, możesz wykonywać bardziej złożone operacje przy użyciu mniejszej liczby wierszy kodu. Na przykład otalSalary +–5 to krótszy sposób zapisu TotalSalary = TotalSalary + 5. Podobnie, TotalSalary –= 5 oznacza, że zmienna TotalSalary zawiera teraz wartość TotalSalary – 5.

UWAGA: Wielu początkujących programistów języka C popełnia błąd, używając pojedynczego znaku równości (=) zamiast podwójnego znaku równości (==) podczas próby sprawdzenia wartości zmiennej. Pojedynczy znak równości (operator przypisania) jest używany do przypisania wartości zmiennej. Na przykład a = 5 przypisuje wartość 5 zmiennej a. Aby sprawdzić wartość zmiennej a, możesz użyć instrukcji if (a == 5). Jeśli omyłkowo napiszesz instrukcję if (a = 5), wartość 5 zostanie przypisana do zmiennej a, a następnie instrukcja zostanie oceniona jako prawdziwa. Dzieje się tak, ponieważ każda wartość różna od zera jest oceniana jako prawdziwa, a wartość zerowa jest oceniana jako fałszywa.

Chociaż ten moduł obejmuje tylko najbardziej podstawowe elementy programu, dzięki temu, czego się do tej pory nauczyłeś, możesz napisać program w C, który wyświetla coś na ekranie. Testerzy bezpieczeństwa powinni zdobyć dodatkowe umiejętności programistyczne, aby móc tworzyć narzędzia do wykonywania określonych zadań, jak zobaczysz w „Understanding Perl" i „Understanding Python" dalej w tym module.

Rozgałęzianie, pętlenie i testowanie w C

Rozgałęzianie w C jest tak proste, jak umieszczenie funkcji w programie, po której następuje średnik. Poniższy kod C nic nie robi, ale pokazuje, jak zacząć pisać program, który można później rozwijać. Na przykład w poniższym kodzie instrukcja prompt(); (oznaczona średnikiem na końcu) na początku rozgałęzia się, aby przejść do funkcji prompt():

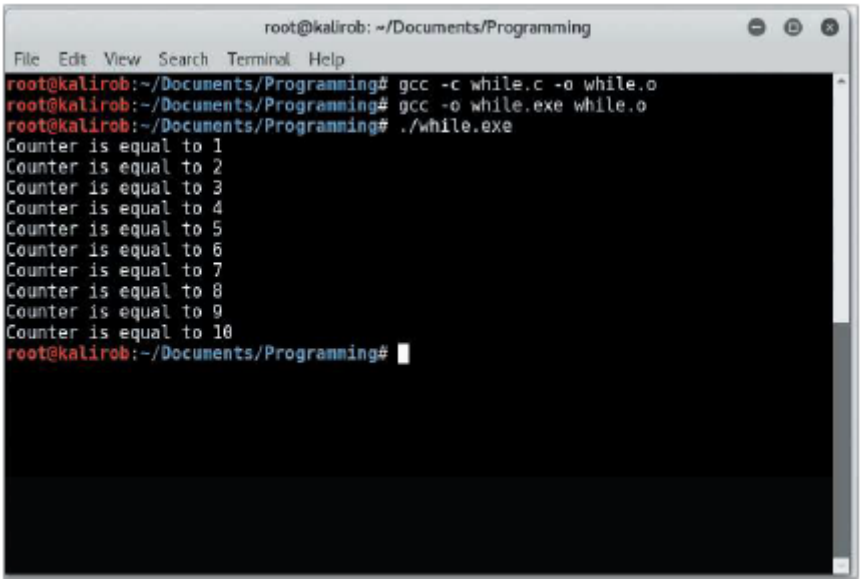
```
main()

{

prompt() ; //Call function to prompt user with a question

display(); //Call function to display graphics onscreen

calculate(); //Call function to do complicated math

cleanup(); //Call function to make all variables equal to

//zero

prompt()

{

[code for prompt() function goes here]

}

display()

{

[code for display() function goes here]

}

[and so forth]

}
```

Po uruchomieniu program rozgałęzia się do funkcji prompt(), a następnie kontynuuje rozgałęzianie do funkcji wymienionych później. Tworząc program w ten sposób, możesz rozwijać każdą funkcję lub moduł po kolei. Możesz również delegować pisanie innych funkcji osobom z większym doświadczeniem w określonych obszarach. Na przykład możesz zlecić czarodziejowi matematyki napisanie funkcji calculator(), jeśli matematyka nie jest twoją mocną stroną. C ma kilka metod pętli. Pętla while to jeden ze sposobów, aby program powtórzył działanie określoną liczbę razy. Sprawdza, czy warunek jest prawdziwy, a następnie kontynuuje pętlenie, aż warunek stanie się fałszywy. Przyjrzyj się poniższemu przykładowi (z pogrubionym ważnym kodem) i sprawdź, czy rozumiesz, co program robi:

```
main()
```

```
{
int counter = 1; //Initialize (assign a value to)
//the counter variable
while (counter <= 10) //Do what's inside the braces until false
{
printf("Counter is equal to %d\n", counter);
++counter; //Increment counter by 1;
}
}
```

Rysunek pokazuje wynik tego programu.



W tym przykładzie, gdy zmienna licznika jest większa niż 10, pętla while zatrzymuje przetwarzanie, co powoduje, że printf() wyświetla 10 wierszy danych wyjściowych przed zatrzymaniem. Pętla do najpierw wykonuje działanie, a następnie sprawdza, czy działanie powinno być kontynuowane. W poniższym przykładzie pętla do najpierw wykonuje funkcję print(), a następnie sprawdza, czy warunek jest prawdziwy:

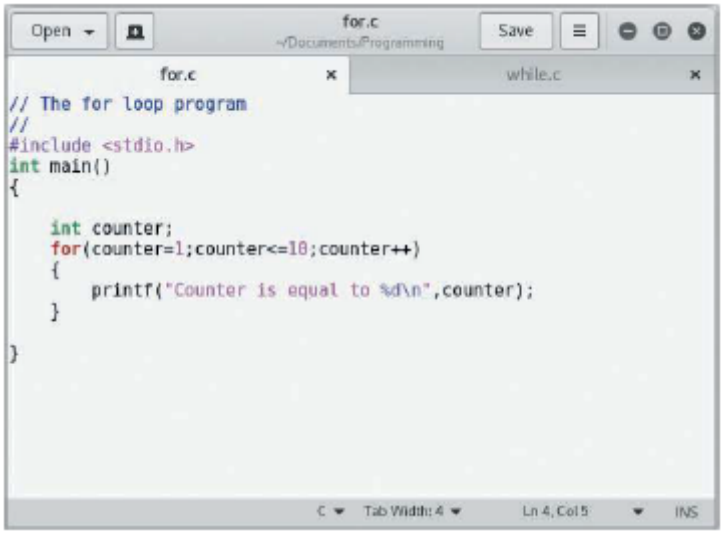
```
main()
{
int counter = 1; //Initialize counter variable
do
{
printf("Counter is equal to %d\n", counter);
++counter; //Increment counter by 1
```

} while (counter <= 10); //Do what's inside the braces until false

}

UWAGA: Co jest lepsze: pętla while czy pętla do? To zależy. Pętla while może nigdy się nie wykonać, jeśli warunek nie jest spełniony. Pętla do zawsze wykonuje się przynajmniej raz.

Ostatnim typem pętli w C jest pętla for, jeden z najciekawszych fragmentów kodu w C. Pętla for zaczyna się od słowa kluczowego for, po którym następują trzy elementy w początkowych i końcowych nawiasach okrągłych (nazywanych również nawiasami okrągłymi). Pierwszy element wewnątrz nawiasów inicjuje zmienną, której użyje pętla for. Drugi element definiuje test, który w przypadku fałszu powoduje wyjście z pętli for. Ostatni element definiuje działanie, które należy wykonać, jeśli test okaże się prawdziwy. Po instrukcji for umieszczasz dowolny kod, który chcesz wykonać, w początkowych i końcowych nawiasach klamrowych (nazywanych również nawiasami klamrowymi). Kod wewnątrz nawiasów klamrowych będzie wykonywany dla każdej iteracji pętli for. Kod pokazany na Rysunku ma następującą pętlę for:



for (counter=1;counter<=10;counter++);

Pierwsza część wewnątrz nawiasów okrągłych inicjuje zmienną całkowitą counter do 1, a następnie druga część testuje warunek. Ten warunek nakazuje pętli for kontynuowanie pętli tak długo, jak zmienna counter ma wartość równą lub mniejszą niż 10. Ostatnia część pętli for zwiększa zmienną counter o 1.

UWAGA: Linia kodu int main() w górnej części programu zaczyna się od int, tak aby funkcja main definiowała typ zwracany. Jeśli pominiesz int, program zostanie pomyślnie skompilowany, ale kompilator wyświetli komunikat ostrzegawczy, że przyjmuje int, ponieważ nie zadeklarowałeś go jawnie.

W niektórych programach w języku C można spotkać pętlę for zawierającą wyłącznie średniki, jak w poniższym przykładzie:

for (;;)

{

printf("Wow!");

}

Ten kod jest potężną, ale niebezpieczną implementacją pętli for. Instrukcja for (;;) mówi kompilatorowi, aby kontynuował wykonywanie tego, co jest w nawiasach, w kółko. Możesz utworzyć nieskończoną pętlę za pomocą tej instrukcji, jeśli nie masz sposobu na wyjście z bloku kodu, który jest uruchomiony. Zwykle programista ma instrukcję wewnątrz bloku, która wykonuje test zmiennej, a następnie wychodzi z bloku, gdy spełniony jest pewien warunek.

Nauka korzystania z kompilatora GNU GCC

Czas trwania: 30 minut

Cel: Nauczenie się korzystania z kompilatora GNU GCC dołączonego do większości systemów operacyjnych *nix.

Opis: W przeszłości programiści musieli czytać kod wiersz po wierszu przed przesłaniem zadania do procesora mainframe. Zadanie obejmowało wszystkie polecenia, które procesor miał wykonać. Jeśli program był pełen błędów, operator komputera mainframe powiadamiał programistę, który musiał ponownie przejrzeć kod i naprawić błędy. Dzięki dzisiejszym kompilatorom możesz napisać program, skompilować go i przetestować samodzielnie. Jeśli kompilator znajdzie błędy, zwykle wskazuje, co to jest, dzięki czemu możesz poprawić kod i ponownie skompilować program. W tym ćwiczeniu tworzysz program w języku C, który zawiera błędy i próbujesz skompilować program. Po zobaczeniu wygenerowanych błędów poprawiasz program, a następnie kompilujesz go ponownie, aż do uzyskania poprawnego wyniku.

1. Uruchom komputer w systemie Kali Linux.

2. Otwórz okno terminala i w wierszu poleceń powłoki wpisz vim syntax.c i naciśnij Enter, aby użyć edytora vim.

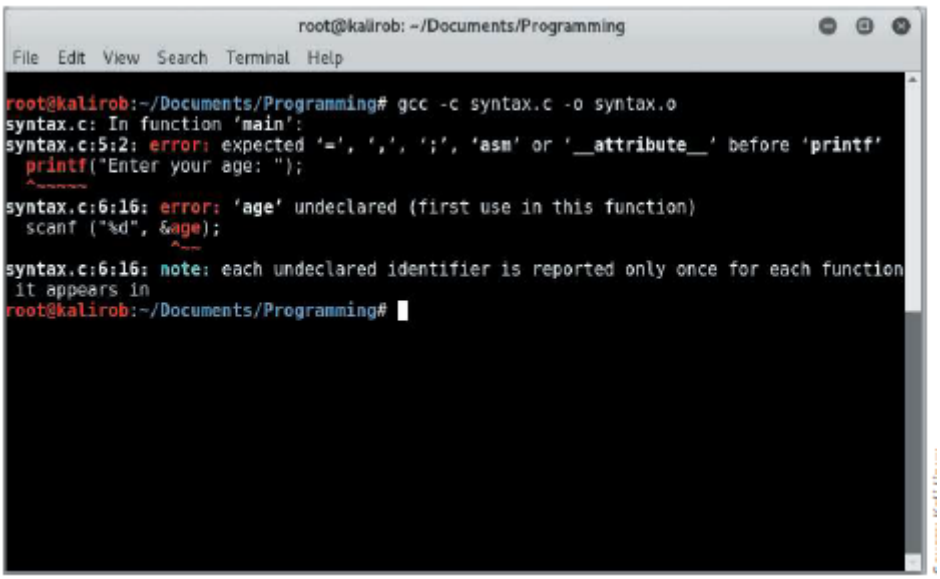
3. Wpisz i, aby przejść do trybu insert.

4. Wpisz następujący kod, naciskając Enter po każdym wierszu:

```
#include <stdio.h>

int main()

{

int age

printf("Enter your age: ");

scanf("%d", &age);

if (age > 0)

{

printf("You are %d years old\n", age);

}

}
```

5. Wyjdź i zapisz plik, naciskając Esc, a następnie : (dwukropek). W wierszu poleceń : wpisz wq i naciśnij Enter.

6. Aby skompilować program, wpisz gcc -o syntax syntax.c i naciśnij Enter. Parametry tego polecenia gcc są uporządkowane inaczej niż w poprzednich przykładach. Ten sposób wykonywania gcc również działa. Przełącznik -o nakazuje kompilatorowi utworzenie pliku wyjściowego o nazwie syntax. Kompilator zwraca błąd (lub kilka błędów) podobny do tego na rysunku



Błąd różni się w zależności od używanej wersji kompilatora. W każdym przypadku powinieneś zostać ostrzeżony o błędzie składni przed printf(). Błąd będzie wskazywał, że kompilator oczekiwał czegoś przed printf(), na przykład znaku równości, przecinka lub średnika. W tym przypadku błąd wystąpił, ponieważ po instrukcji int age nie ma średnika.

UWAGA: Jeśli utworzony przez Ciebie kod źródłowy nie zawiera błędów, wyświetlany jest monit powłoki.

UWAGA: Czasami możesz łatwo naprawić błąd, patrząc na numer wiersza pierwszego wykrytego błędu.

7. Aby poprawić błąd brakującego średnika, możesz ponownie użyć edytora vim. Wpisz vim syntax.c i naciśnij Enter. Wpisz a, aby przejść do trybu dodawania. Dodaj średnik na końcu wiersza zawierającego deklarację zmiennej int age.

8. Zapisz i wyjdź z programu.

9. Skompiluj program ponownie, wpisując gcc -o syntax syntax.c i naciskając Enter. (Możesz również użyć klawisza strzałki w górę, aby powrócić do poprzednich poleceń.)

10. Jeśli wszystko wpisałeś poprawnie, powinien zostać wyświetlony monit powłoki. Aby uruchomić program, wpisz ./syntax i naciśnij Enter.

11. Wyloguj się z sesji Kali Linux na czas następnej aktywności.

UWAGA: Powinieneś wiedzieć, jak używać edytora vi (vim). Nawet jeśli vim może wydawać się prymitywny i niezręczny, w niektórych przypadkach może być jedynym dostępnym edytorem. Edytor gedit jest bardziej przyjazny dla użytkownika i używa kolorów do wyróżniania poleceń, zmiennych i

innych konstrukcji. Podczas kodowania możesz używać gedit zamiast vim, ale ćwicz używanie vim tak często, jak możesz.

BAJTY BEZPIECZEŃSTWA

Istnieją dwie szkoły myślenia na temat tego, jak radzić sobie z błędami składniowymi. Wielu programistów uważa, że kompilator powinien sprawdzać błędy w ich kodzie, więc poświęcają niewiele czasu na czytanie i przechodzenie przez swoje programy w poszukiwaniu błędów składniowych lub logicznych. Po prostu kompilują program i patrzą, jakie błędy się pojawiają. Inni odmawiają kompilacji programu, dopóki nie zbadają dokładnie kodu i nie upewnią się, że jest dokładny i poprawny składniowo. W przypadku początkujących programistów dokładne zbadanie kodu przed kompilacją pomaga stać się lepszym programistą. Zwiększysz swoje umiejętności i rozwiniesz bystre oko potrzebne do zauważenia brakującego nawiasu klamrowego lub średnika

ROZUMIENIE PODSTAW HTML

HTML to język znaczników używany głównie do określania formatowania i układu stron internetowych, więc pliki HTML nie zawierają rodzaju kodu programistycznego, który widzisz w programie C. Jako specjalista ds. bezpieczeństwa powinieneś rozumieć podstawową składnię HTML, ponieważ nadal jest ona podstawą tworzenia stron internetowych. Niezależnie od tego, w jakim języku utworzono stronę internetową, sama strona internetowa zawiera instrukcje HTML, więc znajomość HTML jest podstawą nauki innych języków internetowych. Specjaliści ds. bezpieczeństwa często muszą badać strony internetowe i rozpoznawać, kiedy coś wygląda podejrzanie. Powinieneś rozumieć ograniczenia HTML, umieć czytać plik HTML i mieć podstawową wiedzę na temat tego, co się dzieje. Ta sekcja nie uczyni z Ciebie programisty stron internetowych, ale wprowadza podstawy HTML, dzięki czemu będziesz mieć podstawę do eksploracji i nauki innych języków programowania i skryptów. Obecną wersją HTML jest HTML5. Oferuje ona znaczące ulepszenia w stosunku do wcześniejszych wersji HTML. Na przykład HTML5 natywnie obsługuje elementy multimedialne, takie jak wideo i audio. Odtwarzacze multimedialne innych firm (takie jak nieistniejący już Flash Player) często sprawiały problemy i stanowiły zagrożenie dla bezpieczeństwa, ale nie są już potrzebne. Możliwość natywnej obsługi bogatych multimediów w HTML5 rozwiązała te problemy.

UWAGA: Obecnie wiele stron internetowych używa Extensible Markup Language (XML). Chociaż ten język nie jest omawiany w tej książce, jest to dobry język do nauki, jeśli chcesz specjalizować się w bezpieczeństwie sieci. Nauka dodatkowych języków programowania sieci, takich jak Extensible HTML (XHTML; zobacz www.w3c.org, aby uzyskać więcej informacji), Perl, JavaScript, PHP i Python, może również zwiększyć Twoje umiejętności jako specjalisty ds. bezpieczeństwa.

Tworzenie strony internetowej w HTML

Możesz utworzyć stronę internetową HTML w Notatniku, a następnie wyświetlić ją w przeglądarce internetowej. Ponieważ HTML jest językiem znaczników, a nie językiem programowania, nie używa rozgałęzień, pętli ani testowania. Poniżej znajduje się prosty przykład kodu HTML:

```
<!-- This is how you add a comment to an HTML webpage -->

<html>

<head>

<title>Hello, world--again</title>

</head>
```

<body>

This is where you put page text, such as marketing copy for an e-commerce business.

</body>

</html>

Symbole < i > oznaczają znaczniki HTML, które działają na zawarte w nich dane. Zauważ, że każdy znacznik ma pasujący znacznik zamykający, który zawiera ukośnik (/). Na przykład znacznik <html> ma znacznik zamykający </html>, a znaczniki <head>, <title> i <body> mają podobne znaczniki zamykające. Większość stron internetowych HTML zawiera te cztery znaczniki. Tabela opisuje niektóre typowe znaczniki formatowania używane na stronie internetowej HTML.

Tag otwierający: Tag zamykający: Opis

<h1>, <h2>, <h3>, <h4>, <h5> i <h6>: <h1>, <h2>, <h3>, </h4>, </h5> i </h6>: Formatuje tekst jako różne poziomy nagłówków. Poziom 1 to największy rozmiar czcionki, a poziom 6 to najmniejszy.

: : Zaznacza początek i koniec akapitu.

: L Formatuje zamknięty tekst pogrubieniem.

: : Formatuje zamknięty tekst kursywą.

Inne znaczniki formatują tabele, listy i inne elementy, ale Tabela daje ogólny przegląd znaczników HTML. W Internecie można znaleźć wiele odniesień, aby dowiedzieć się więcej o tworzeniu stron internetowych HTML. W Ćwiczeniu 3 ćwiczysz tworzenie strony internetowej przy użyciu Notatnika jako edytora

Tworzenie strony internetowej HTML

Czas trwania: 30 minut

Cel: Utwórz stronę internetową HTML.

Opis: Jako tester bezpieczeństwa możesz być zobowiązany do przeglądania stron internetowych w celu sprawdzenia możliwych problemów z bezpieczeństwem sieci. Podstawowa znajomość języka HTML może pomóc Ci w tym zadaniu. W tej aktywności utworzysz prostą stronę internetową HTML, a następnie wyświetlisz ją w przeglądarce.

1. Uruchom komputer w systemie Windows. W systemie Windows 10 kliknij prawym przyciskiem myszy przycisk Start, kliknij polecenie Uruchom, wpisz notepad MyWeb.html, a następnie naciśnij klawisz Enter. Jeśli zostaniesz poproszony o utworzenie nowego pliku, kliknij przycisk Tak.

2. W nowym dokumencie Notatnika wpisz następujące wiersze, naciskając klawisz Enter po każdym wierszu:

<!-- Ta strona internetowa HTML ma wiele tagów -->

<html>

<head>

<title>HTML dla testerów zabezpieczeń</title>

</head>

3. Wpisz kolejne dwa wiersze, naciskając klawisz Enter dwa razy po każdym wierszu:

<body>

Strona internetowa testera zabezpieczeń

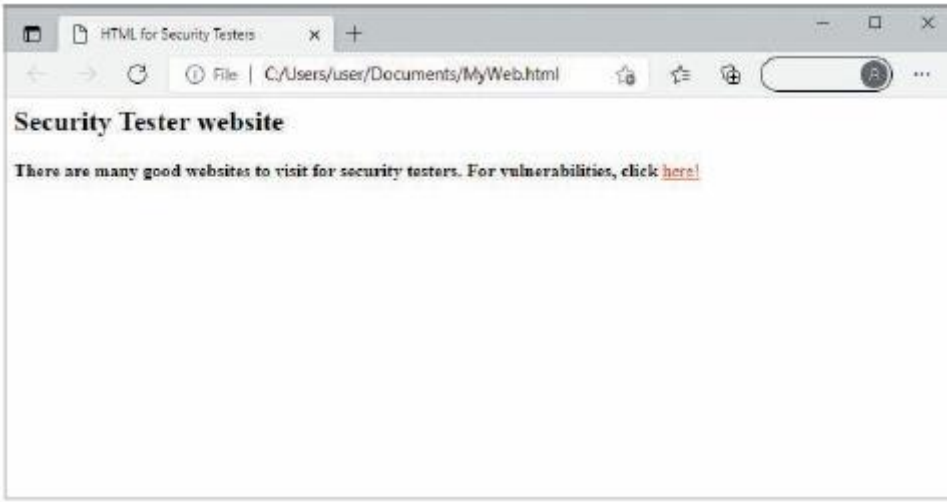
4. Wpisz Istnieje wiele dobrych witryn internetowych dla testerów zabezpieczeń. W przypadku luk kliknij i naciśnij klawisz Enter.

5. Wpisz tutaj! i naciśnij klawisz Enter.

6. Wpisz i naciśnij klawisz Enter. 7. Wpisz </body> i naciśnij Enter. W ostatnim wierszu wpisz </html>, aby zakończyć kod.

8. Sprawdź, czy wszystko wpisałeś poprawnie. Po zakończeniu zapisz plik.

9. Aby sprawdzić, czy poprawnie utworzyłeś stronę internetową, uruchom Eksplorator plików i przejdź do domyślnej lokalizacji — zwykle C:\Users\YourUserName\Documents). Kliknij prawym przyciskiem myszy utworzony plik MyWeb.html, wskaż polecenie Otwórz za pomocą, a następnie kliknij Microsoft Edge. Jeśli poprawnie wpisałeś informacje, Twoja strona internetowa powinna wyglądać tak, jak pokazano na rysunku .



10. Kliknij hiperłącze here!, które utworzyłeś, aby sprawdzić, czy zostałeś przekierowany do właściwej witryny. Jeśli nie, wprowadź poprawki do kodu HTML.

11. Po zakończeniu zamknij przeglądarkę, ale pozostaw system Windows uruchomiony do następnej czynności.

ZROZUMIENIE PERL

Wiele skryptów i programów dla profesjonalistów ds. bezpieczeństwa jest napisanych w Practical Extraction and Report Language (Perl), potężnym języku skryptowym. Perl i Python to dwa popularne języki dla profesjonalistów ds. bezpieczeństwa; ten moduł obejmuje podstawy Perla. W tej sekcji dowiesz się, dlaczego ten język jest tak popularny, przeanalizujesz składnię języka i przećwiczysz pisanie skryptów Perla. Utworzysz również narzędzie do badania konfiguracji komputera z systemem Windows.

Informacje ogólne o Perlu

Perl, opracowany przez Larry'ego Walla w 1987 roku, może działać na niemal każdej platformie, a systemy operacyjne oparte na *nix mają niezmiennie zainstalowany Perl. Składnia Perla jest podobna do C, więc programiści C nie mają większych trudności z nauką Perla. Od czasu jego powstania wydano ponad 20 wersji Perla. Każda nowa wersja jest tworzona i wydawana w celu naprawienia błędów, dodania nowych funkcji i rozwiązania problemów bezpieczeństwa. Niektóre aktualizacje obejmują duże rewizje całego systemu Perl. Perl 5.34 to aktualna stabilna wersja, która została wydana w maju 2021 r. Więcej szczegółów znajdziesz na stronie https://en.wikipedia.org/wiki/Perl. Hakerzy używają Perla do tworzenia zautomatyzowanych exploitów i złośliwych botów, ale administratorzy systemów i specjaliści ds. bezpieczeństwa używają go do wykonywania powtarzalnych zadań i monitorowania bezpieczeństwa. Przed zapoznaniem się ze składnią Perla napiszesz swój pierwszy skrypt Perla w Zadaniu 4. Jak w przypadku każdego języka programowania, najlepszym sposobem na naukę Perla jest jego używanie.

Pisanie skryptu Perla przy użyciu GVim

Czas trwania: 60 minut

Cel: Napisz skrypt Perla przy użyciu GVim.

Opis: Zarówno specjaliści ds. bezpieczeństwa, jak i hakerzy używają języka skryptowego Perl. Wiele programów hakerskich jest napisanych w Perlu, więc wszelkie umiejętności, które rozwiniesz w tym języku, pomogą Ci w Twojej karierze. W poprzednich ćwiczeniach używałeś vim jako edytora tekstu. W tym ćwiczeniu napiszesz podstawowy skrypt Perla przy użyciu graficznej wersji vim, zwanej GVim. W przeciwieństwie do vim, GVim pozwala na nawigację za pomocą klikania zamiast polegać na poleceniach klawiatury.

1. Uruchom komputer w systemie Kali Linux.

2. Otwórz okno terminala, a następnie zmień katalog na pulpit za pomocą polecenia cd ~/Desktop.

3. Wpisz gvim first.pl i naciśnij Enter.

4. Wybierz kartę Składnia u góry okna GVim i wybierz opcję Automatyczne. Włączy to podświetlanie składni dla Twojego projektu Perl.

5. W pierwszym wierszu wpisz # this is my first Perl script program i naciśnij Enter.

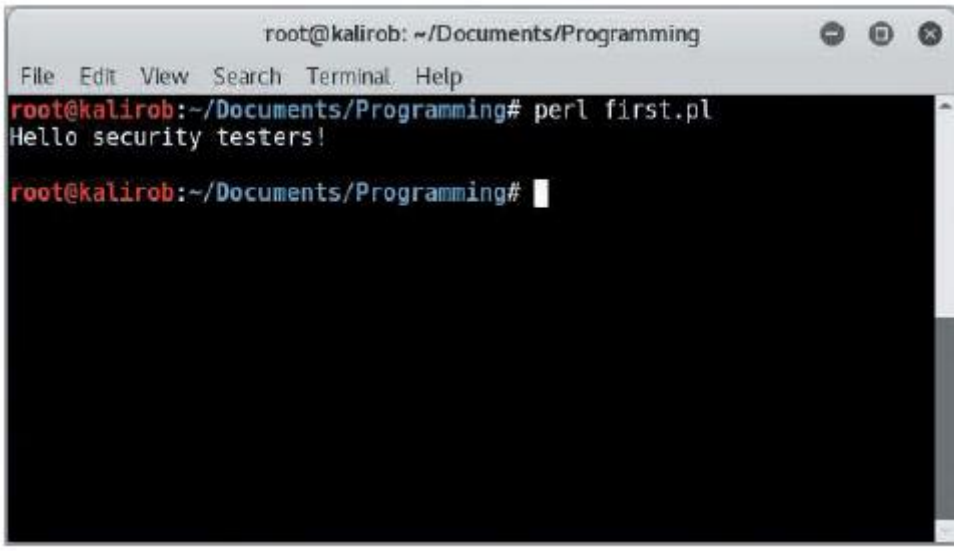
6. Wpisz # I should always have documentation/comments in my scripts! i naciśnij Enter dwa razy.

7. Wpisz #This code display "Hello security testers" on screen i naciśnij Enter dwa razy, aby dodać kolejny komentarz opisujący, co robi poniższy kod.

8. Wpisz print "Hello security testers!\n\n"; i naciśnij Enter. Uważaj, aby nie pominąć średnika lub cudzysłowu. Pamiętaj, że programowanie wymaga bystrego oka.

9. Zapisz plik.

10. W wierszu poleceń wpisz perl first.pl i naciśnij Enter. Jeśli kod nie zawiera błędów, ekran powinien wyglądać jak na rysunku.



Jeśli otrzymasz komunikaty o błędach, przeczytaj plik i porównaj go z wierszami kodu w krokach tej aktywności. Popraw wszelkie błędy i zapisz plik ponownie.

11. Zamknij okno wiersza poleceń i pozostaw system Windows uruchomiony do następnej aktywności.

Zrozumienie podstaw języka Perl

Przydatna jest wiedza, jak szybko uzyskać pomoc w dowolnym języku programowania. Polecenie perl -h podaje listę parametrów używanych z poleceniem perl. Witryna internetowa https://perldoc.perl.org/ jest doskonałym repozytorium informacji o języku Perl. Możesz wybrać wersję języka Perl, która Cię interesuje, a następnie wyszukać informacje dotyczące tej wersji, takie jak szczegółowe opisy poleceń. Witryna internetowa zawiera również samouczki i często zadawane pytania (FAQ), do których możesz uzyskać dostęp, aby dowiedzieć się więcej. Perl ma polecenie printf do formatowania złożonych zmiennych. Tabela 7-8 pokazuje, jak używać tego polecenia do formatowania określonych danych. Zwróć uwagę na podobieństwa do języka C.

Znak formatujący : Opis : Wejście : Wyjście

%c : Znak printf : ‘%c’ , "d" : d

%s : Ciąg : printf ‘%s’, "To jest fajne!" : To jest fajne!

%d : Liczba całkowita ze znakiem w systemie dziesiętnym : printf ‘%+d%d’, 1, 1 : +1 1

%u : Liczba całkowita bez znaku w systemie dziesiętnym : printf ‘%u’, 2 : 2

%o : Liczba całkowita bez znaku w systemie ósemkowym : printf ‘%o’ , 8 : 10

%x : Liczba całkowita bez znaku w systemie szesnastkowym : printf ‘%x’, 10 : a

%e : Liczba zmiennoprzecinkowa w notacji naukowej : printf ‘%e’ , 10; : 1.000000e+001 (w zależności od systemu operacyjnego)

%f : Liczba zmiennoprzecinkowa w notacji dziesiętnej stałej : printf ‘%f’ , 1; : 1.000000

Zrozumienie BLT w Perlu

Jak się wcześniej dowiedziałeś, wszystkie języki programowania muszą mieć sposób na rozgałęzianie, pętlenie i testowanie. Poniższe sekcje wykorzystują przykłady kodu, aby pokazać, jak Perl obsługuje te funkcje BLT. Podczas analizowania tych przykładów pamiętaj o następujących zasadach składni:

• Słowo kluczowe sub jest używane przed nazwami funkcji.

• Zmienne zaczynają się od symbolu $.

• Wiersze komentarzy zaczynają się od symbolu #.

• Symbol & oznacza funkcję.

Poza tymi drobnymi różnicami składnia Perla jest bardzo podobna do składni języka C. To podobieństwo jest jednym z powodów, dla których wielu specjalistów ds. bezpieczeństwa z doświadczeniem w programowaniu w języku C wybiera Perl jako język skryptowy.

Rozgałęzienie w Perlu

W programie Perl, aby przejść z jednej funkcji do drugiej, wywołujesz funkcję, wpisując jej nazwę w kodzie źródłowym. W poniższym przykładzie linia &name_best_guitarist rozgałęzia program do podfunkcji name_ best_guitarist:

```
# Perl program illustrating the branching function

# Documentation is important

# Initialize variables

$first_name = "Jimi";

$last_name = "Hendrix";

&name_best_guitarist;

sub name_best_guitarist

{

printf "%s %s %s", $first_name, $last_name, "was the best!";

}
```

Pętla w Perlu

Załóżmy, że chcesz wysłać ważną wiadomość do wszystkich w swojej klasie, używając polecenia Net send. Ponieważ wysyłasz tę samą wiadomość do wielu użytkowników, jest to powtarzalne zadanie, które wymaga pętli. W Ćwiczeniu 7-5 piszesz skrypt Perla, który właśnie to robi: wysyła wiadomość do wszystkich w klasie. Jak się nauczyłeś w C, masz kilka możliwości wykonania pętli. W tej sekcji poznasz dwa mechanizmy pętli Perla: pętlę for i pętlę while. Pętla for w Perlu jest identyczna z pętlą for w C:

```
for (variable assignment; test condition; increment variable)

{

a task to do over and over

}
```

Substituting the variable $a, you have the following code:

```
for ($a = 1; $a <= 10; $a++)

{

print "Hello, security testers!\n"

}
```

Ta pętla drukuje frazę 10 razy. Następnie spróbuj wyprodukować ten sam wynik za pomocą pętli while, która ma następującą składnię:

```
while (test condition)

{

a task to do over and over

}
```

Poniższy kod generuje taki sam wynik jak pętla for:

```
$a = 1;

while ($a <= 10)

{

print "Hello, security testers!\n";

$a++

}
```

BAJTY BEZPIECZEŃSTWA

Chris Nandor, znany z opracowania wersji Mac Classic Perl 5.8.0, został jednym z pierwszych hakerów, którzy użyli skryptu Perla w wyborach online. Najwyraźniej jego skrypt Perla dodał ponad 40 000 głosów dla kilku graczy Red Sox podczas wyborów online w 1999 r. na mecz All-Stars. Podobnie, w 1993 r. wybory online z udziałem Denver Broncos wyśledziły ponad 70 000 głosów pochodzących z jednego adresu IP. Siła pętli!

Testowanie warunków w Perlu

Większość programów musi być w stanie przetestować wartość zmiennej lub warunku. Dwa poprzednie przykłady pętli używają operatora mniejszego lub równego (<=). Inne operatory używane do testowania w Perlu są podobne do operatorów języka C. Tabela zawiera listę operatorów, których można używać w Perlu.

Operator : Funkcja : Przykład

+ : Dodawanie : $total = $sal + $commission

– : Odejmowanie : $profit = $gross sales – $cost of goods

* : Mnożenie : $total = $cost * $quantity

/ : Dzielenie : $GPA = $total_points / $number of classes

% : Moduł : $a % 10 = 1

** : Wykładnik : $total = $a**10

Zadania

= : Zadanie : $Last name = "Rivera"

+= ; Dodaj, następnie przypisanie : $a+ = 10; skrót dla $a=$a+10

-= : Odejmij, następnie przypisanie : $a–=10; skrót dla $a=$a–10

= : Pomnóż, następnie przypisanie : $a = 10; skrót dla $a=$a*10

/= : Dzielenie, następnie przypisanie : $a/ = 10; skrót dla $a=$a/10

%= : Moduł, następnie przypisanie : $a%=10; skrót dla $a=$a%10

= : Wykładnik i przypisanie : $a=2; skrót dla $a=$a**2

++ ": Inkrementacja: $a++; inkrementacja $a o 1

— : Dekrementacja: a—; dekrementacja $a o 1

Porównania

== : Równe: $a==1; porównaj wartość $a z 1

!= : Nierówne: $a!=1; $a nie jest równe 1

> : Większe niż: $a>10

< : Mniejsze niż: $a<10

>= : Większe niż lub : do $a>=10

<= : Mniejsze lub równe: $a<=10

Często łączy się te operatory z warunkami Perla, takimi jak poniższe:

• if — sprawdza, czy warunek jest prawdziwy. Przykład:

if ($age < 12) {

print "You must be a know-it-all!";

}

• else — używane, gdy istnieje tylko jedna opcja do wykonania, jeśli warunek nie jest prawdziwy. Przykład:

if ($age) > 12 {

print "You must be a know-it-all!";

}

else

{

```
print "Sorry, but I don’t know why the sky is blue.";
}
```

• elsif — używane, gdy istnieje kilka warunków do przetestowania. Przykład:

```
if ( ($age > 12) && ($age < 20) )
{
print "You must be a know-it-all!";
}
elsif ($age > 39)
{
print "You must kłamać o swoim wieku!";
}
else
{
print "Być młodym...";
}
```

• unless—Wykonuje się, chyba że warunek jest prawdziwy. Przykład:

```
unless ($age == 100)
{
print "Jeszcze wystarczająco dużo czasu, aby uzyskać tytuł licencjata.";
}
```

Wiadomość jest wyświetlana, dopóki zmienna $age nie będzie równa 100. Przy odrobinie praktyki i dużej cierpliwości, te przykłady mogą dać ci początek tworzenia funkcjonalnych skryptów Perl.

Pisanie skryptu Perla do testowania bezpieczeństwa

Czas trwania: 30 minut

Cel: Napisz skrypt Perla, który używa komponentów rozgałęziających, pętli i testujących.

Opis: Specjaliści ds. bezpieczeństwa często muszą automatyzować lub tworzyć narzędzia, które pomogą im przeprowadzać testy bezpieczeństwa. W tej aktywności napiszesz skrypt Perla, który używa polecenia notification-send i pętli for, aby wybrać numery IP z zakresu klasy podanego przez instruktora. Możesz użyć następującego odniesienia do polecenia Perl ping: http://perldoc.perl.org/Net/Ping.html.

1. Zapisz zakres adresów IP używany w sieci klasy.

2. Otwórz okno terminala, wpisz apt-get install libnotify-bin, a następnie naciśnij Enter, aby zainstalować usługę powiadomień, której będziesz używać w tej aktywności.

3. Zmień katalog na pulpit za pomocą cd ~/Desktop. Aby użyć edytora gedit do utworzenia skryptu, wpisz gedit ping.pl i naciśnij Enter.

4. Wpisz # Program do pingowania stacji roboczych w sieci i naciśnij Enter.

5. Wpisz # Jeśli ping się powiedzie, na ekranie zostanie wysłana wiadomość i naciśnij Enter.

6. Wpisz # Program przyjmuje adres klasy c (w.x.y.z), gdzie w.x.y jest częścią sieciową adresu i naciśnij Enter.

7. Wpisz # zmień wartość $class_iP poniżej, aby odzwierciedlała część sieciową Twojej sieci i naciśnij Enter.

8. Wpisz # Oktet „z” zostanie zwiększony z 1 do 254 za pomocą pętli for i naciśnij Enter dwa razy. 9. Wpisz use diagnostics; i naciśnij Enter dwa razy.

10. Wpisz use net::Ping; # ładuje bibliotekę net i naciśnij Enter dwa razy.

11. Wpisz $p = net::Ping->new(); # tworzy nowy obiekt ping z domyślnymi ustawieniami i naciśnij Enter dwa razy.

12. Następny wiersz inicjuje zmienną, której używasz do przechowywania identyfikatora sieci. Wpisz $class_iP = "192.168.2"; # Identyfikator sieci (zmień wartość w cudzysłowie, aby odzwierciedlała topologię) i naciśnij Enter dwa razy. Następne wiersze kodu to pętla for, która zwiększa ostatni oktet adresu IP sieci do wszystkich dostępnych adresów IP w Twojej klasie. Kod wewnątrz pętli for powinien być wcięty dla czytelności.

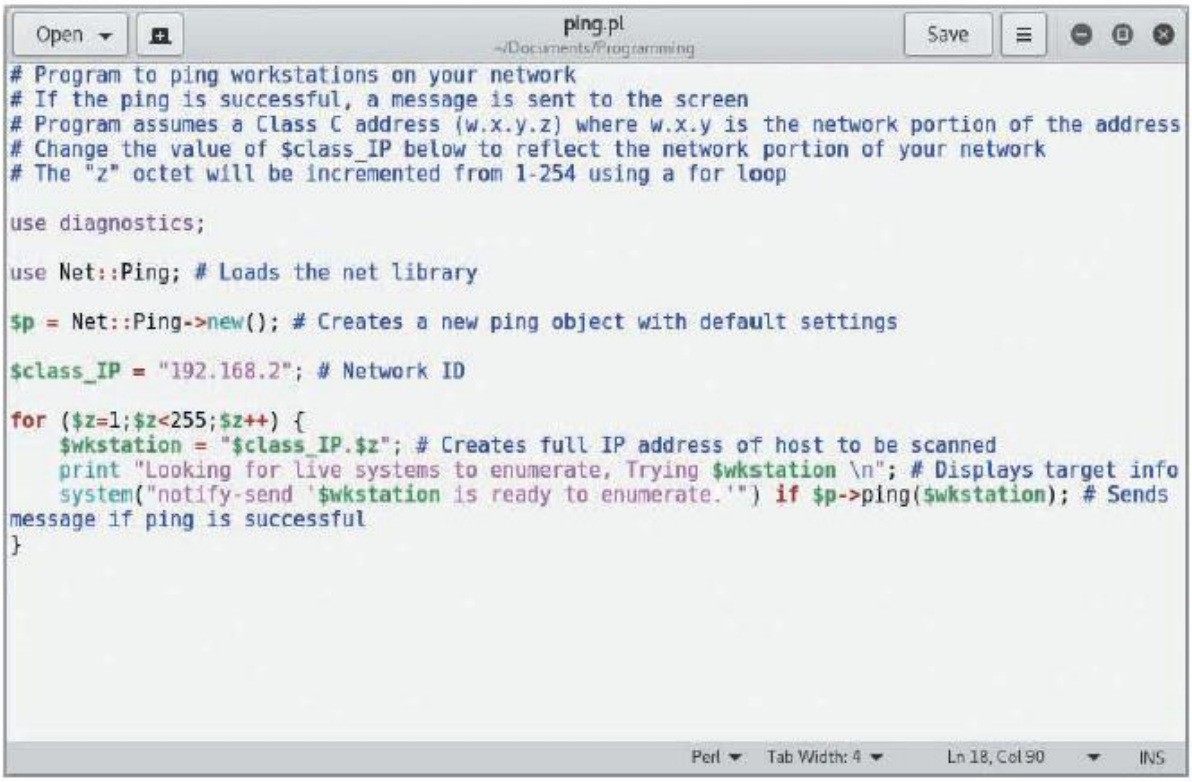
13. Wpisz for ($z=1; $z<255; $z++) { i naciśnij Enter.

14. Naciśnij klawisz Tab, aby wciąć, wpisz $wkstation = "$class_iP.$z"; # tworzy pełny adres IP hosta, który ma zostać przeskanowany, a następnie naciśnij Enter.

15. Naciśnij klawisz Tab, aby wciąć, wpisz print "Looking for live systems to enumerate. Trying $wkstation \n"; # Wyświetla informacje o celu, a następnie naciśnij Enter.

16. Naciśnij klawisz Tab, aby wciąć, wpisz system("notify-send ‘$wkstation is ready to enumerate.’") if $p->ping($wkstation); # wysyła wiadomość, jeśli ping się powiedzie, a następnie naciśnij Enter.

17. Wpisz } i naciśnij Enter, aby zakończyć program, który powinien wyglądać podobnie do Rysunku.



18. Aby ulepszyć dokumentację tego programu, dodaj wiersze komentarzy do swojego kodu, podając autora i datę napisania oraz wyjaśniając wszelkie złożone algorytmy.

19. Przejdź przez każdy wiersz kodu i upewnij się, że składnia jest poprawna. Zwróć uwagę, że zmienna $class_IP zawiera część sieciową zakresu adresów IP Twojej klasy. Po sprawdzeniu składni i zawartości skryptu Perl, zapisz go i wróć do okna terminala.

20. Uruchom skrypt, wpisując perl ping.pl i naciskając Enter. Jeśli nie ma błędów, program powinien rozpocząć pingowanie adresów IP. Jeśli zostanie znaleziony aktywny adres, zobaczysz powiadomienie na dole ekranu.

21. Aby zakończyć skrypt Perl, naciśnij ctrl+c. Pozostaw okno wiersza poleceń otwarte do następnej aktywności.

ZROZUMIENIE KONCEPCJI PROGRAMOWANIA OBIEKTOWEGO

Właśnie wtedy, gdy myślisz, że jesteś już zadowolony z koncepcji technologicznej, pojawia się coś nowego. Chociaż koncepcja programowania obiektowego (OOP) nie jest nowa dla doświadczonych programistów, może nie być znana tym, którzy dopiero uczą się pisać swój pierwszy skrypt Perla. Perl 5.0 używa koncepcji programowania obiektowego, a Perl 6.0 będzie oparty wyłącznie na tym modelu, więc ta sekcja obejmuje kilka podstawowych koncepcji obiektowych jako podstawę do napisania kolejnego skryptu Perla. Ta sekcja w żadnym wypadku nie jest kompletnym omówieniem złożonej koncepcji. Nauka programowania obiektowego wymaga czasu i praktyki, a ta sekcja jedynie wprowadza Cię do podstawowych koncepcji.

Składniki programowania obiektowego

Wersja Perla, którą zainstalowałeś, ma dodatkowe funkcje, które mogą wykonywać wywołania programu do interfejsu programowania aplikacji Windows (Win API). Programiści powinni wiedzieć, jakie funkcje są dostępne w różnych systemach operacyjnych, aby mogli pisać programy, które

wchodzą w interakcję z tymi funkcjami. Na przykład programista C wie, że Win API ma funkcję NodeName(), która zwraca nazwę komputera NetBIOS. Aby użyć tej funkcji, programista odwołuje się do niej za pomocą Win32::NodeName(). :: oddziela nazwę klasy, Win32, od funkcji członkowskiej, NodeName(). W programowaniu obiektowym klasy są strukturami, które przechowują fragmenty danych i funkcje. Poniższy przykład kodu pokazuje klasę Employee w C++. Klasy można pisać w wielu językach obiektowych (np. Java, Object COBOL i Perl). Ważne jest, aby wiedzieć, jak wygląda klasa:

```
// This is a class called Employee created in C++

class Employee

{

public:

char firstname[25];

char lastname[25];

char PlaceOfBirth[30];

[code continues]

};

void GetEmp()

{

// Perform tasks to get employee info

[program code goes here]

}
```

Struktura utworzona w tym kodzie może zawierać informacje o pracownikach, a także funkcję, która wykonuje wyszukiwanie. Funkcja zawarta w klasie jest nazywana funkcją członkowską. Jak wspomniano, aby uzyskać dostęp do funkcji członkowskiej, należy użyć nazwy klasy, po której następują dwa dwukropki i nazwa funkcji członkowskiej:

```
Employee::GetEmp()
```

Klasa Win32 zawiera wiele funkcji, które można wywołać ze skryptu Perl. Tabela 7-10 opisuje niektóre powszechnie używane funkcje API Win32.

Funkcja: Opis

GetLastError() : Zwraca ostatni błąd wygenerowany podczas wywołania Win32 API.

OLELastError() : Zwraca ostatni błąd wygenerowany przez Object Linking and Embedding (OLE) API.

BuildNumber() : Zwraca numer kompilacji Perl.

LoginName() : Zwraca nazwę użytkownika osoby uruchamiającej Perl.

NodeName() : Zwraca nazwę komputera NetBIOS.

DomainName() : Zwraca nazwę domeny, do której należy komputer.

FsType() : Zwraca nazwę systemu plików, takiego jak NTFS lub FAT.

GetCwd() : Zwraca bieżący aktywny dysk.

SetCwd(newdir) : Umożliwia zmianę na dysk oznaczony zmienną newdir.

GetOSName() : Zwraca nazwę systemu operacyjnego.

FormatMessage(error) : Konwertuje numer komunikatu o błędzie na opisowy ciąg znaków.

Spawn(command, args, $pid) : Uruchamia nowy proces, używając argumentów dostarczonych przez programistę i identyfikatora procesu ($pid).

LookupAccountSID(sys, sid, $acct, $domain, $type) : Zwraca nazwę konta, nazwę domeny i typ identyfikatora zabezpieczeń (SID).

InitiateSystemShutdown(machine, message, timeout, forceclose, reboot) : Zamyka określony komputer lub serwer Opis funkcji

AbortSystemShutdown(machine) : Przerywa zamykanie, jeśli zostało wykonane przez błąd.

GetTickCount() : Zwraca liczbę taktów Win32 (czas, jaki upłynął od pierwszego uruchomienia systemu).

ExpandEnvironmentalStrings (envstring) : Zwraca ciągi zmiennych środowiskowych określone w zmiennej envstring.

GetShortPathName(longpathname) : Zwraca wersję 8.3 długiej ścieżki. W programach DOS i starszych systemach Windows nazwy plików mogły mieć tylko osiem znaków, z rozszerzeniem trzyznakowym.

GetNextAvailableDrive() : Zwraca następną dostępną literę dysku.

RegisterServer(libraryname) : Ładuje bibliotekę DLL określoną przez libraryname i wywołuje funkcję DLLRegisterServer().

UnregisterServer(libraryname) : Ładuje bibliotekę DLL określoną przez libraryname i wywołuje funkcję DLLUnregisterServer().

Sleep(time) : Wstrzymuje liczbę milisekund określoną przez zmienną time.

Atakujący i specjaliści ds. bezpieczeństwa mogą używać tych funkcji do odkrywania informacji o zdalnym komputerze. Chociaż te funkcje nie są trudne do zrozumienia, nabycie biegłości w ich używaniu w programie wymaga czasu i dyscypliny. Dla specjalistów ds. bezpieczeństwa, którzy muszą wiedzieć, co potrafią atakujący, zdobycie tej umiejętności jest warte poświęconego czasu i wysiłku. W Ćwiczeniu 7-6 tworzysz skrypt Perla, który używa niektórych funkcji API Win32 wymienionych w Tabeli. Ten skrypt podaje następujące informacje o komputerze z systemem Windows, którego używałeś do działań tego modułu:

• Nazwa logowania użytkownika

• Nazwa komputera

• System plików

• Bieżący katalog

• Nazwa systemu operacyjnego

Tworzenie skryptu Perl w celu uzyskania dostępu do interfejsu API Win32

Czas trwania: 30 minut

Cel: Zainstaluj Perl w systemie Windows i dowiedz się, jak uzyskać dostęp do interfejsu API Win32 ze skryptu Perl.

Opis: W tej aktywności zainstalujesz ActivePerl na komputerze z systemem Windows i napiszesz podstawowy skrypt Perl, używając poznanych już funkcji formatowania i funkcji interfejsu API Win32 w tabeli. System Windows domyślnie nie obsługuje języka Perl, więc musisz pobrać i zainstalować silnik Perl. Jeśli to możliwe, pracuj w grupach trzy- lub czteroosobowych. Uruchom przeglądarkę internetową i przejdź do witryny www.activestate.com/products/perl/.

1. Strona ActivePerl wyświetla wersję bezpłatną i komercyjną. Kliknij przycisk POBIERZ, aby pobrać wersję bezpłatną. Kliknięcie przycisku POBIERZ wyświetla stronę pobierania. Przewiń w dół, a następnie kliknij łącze activePerl 5.28 dla systemu Windows, aby pobrać Perl. W miarę dostępności nowszych wersji ActivePerl numer wersji wzrasta, więc może nie być 5.28 podczas pobierania.

2. Po pobraniu pliku zlokalizuj i uruchom plik instalacyjny. W razie potrzeby odpowiedz na wszelkie monity bezpieczeństwa.

3. W oknie powitalnym Kreatora instalacji ActivePerl kliknij Dalej.

4. Przeczytaj umowę licencyjną, sprawdź, czy wybrano przycisk opcji Akceptuję warunki Umowy licencyjnej, a następnie kliknij Dalej.

5. Jeśli musisz zmienić domyślną lokalizację, wprowadź ścieżkę do nowej lokalizacji. Kliknij Dalej.

6. W oknie Gotowy do instalacji kliknij Zainstaluj. Kliknij Tak, aby odpowiedzieć na monit UAC, jeśli to konieczne. Po kilku minutach program zostanie zainstalowany.

7. W ostatnim oknie kliknij Zakończ. Przeczytaj notatki dotyczące wydania, które zostaną automatycznie wyświetlone w przeglądarce internetowej. Następnie zainstaluj program o nazwie Notepad++, który będzie służył jako edytor programowania. Przypomnij sobie, że Notepad++ jest ulepszeniem programu Notatnik systemu Windows. Notepad++ jest świadomy konstrukcji programistycznych i wyświetla kod w formacie, który pomaga podczas kodowania.

8. Przejdź do witryny https://notepad-plus-plus.org/ i pobierz oraz zainstaluj najnowszą wersję. Pobierz 64-bitową (x64) wersję instalatora.

9. Aby rozpocząć pisanie skryptu Perl, uruchom aplikację Notepad++. Zapisz plik jako Win32.pl w katalogu C:\Perl64.

10. W nowym pliku Notepad++ wpisz # Win32.pl w pierwszym wierszu i naciśnij Enter.

11. Wykorzystaj wiedzę zdobytą w tym module do napisania komentarzy w celu udokumentowania programu. Pamiętaj, aby wpisać nazwisko autora, datę i krótki opis tego, co program robi, na przykład funkcje, do których uzyskuje dostęp z Win32 API.

12. Po wierszach dokumentacji naciśnij Enter kilka razy, aby utworzyć puste wiersze oddzielające komentarze od kodu programu. Następnie wpisz use win32; i naciśnij Enter. (Uwaga: nie zapomnij o średniku.)

13. Potrzebujesz pięciu informacji (wymienionych na liście wypunktowanej przed tą czynnością) z Win32 API. Spróbuj napisać kod, aby uzyskać te informacje, a następnie zapisz program. Jeśli potrzebujesz pomocy, wykonaj następujące kroki.

14. Wpisz $login = win32::loginname(); i naciśnij Enter. Ten wiersz wypełnia zmienną $login informacjami zebranymi z LoginName().

15. Wpisz następujące wiersze, aby wypełnić inne zmienne potrzebne do wykonania zadania, naciskając Enter po każdym wierszu:

```
$NetBIOS = Win32::NodeName();

$Filesystem = Win32::FsType();

$Directory = Win32::GetCwd();

$os_name = Win32::GetOSName();
```

16. Następujące zmienne muszą zostać wyświetlone na ekranie. Wpisz wiersze kodu, jak pokazano, naciskając Enter po każdym wierszu.

```
print "$login\n";

print "$NetBIOS\n";

print "$Filesystem\n";

print "$Directory\n";

print "$os_name\n";
```

17. Po wpisaniu całego kodu zapisz program, uruchom go i debuguj wszelkie błędy. Rysunek 7-15 przedstawia wynik. Co jest nie tak z tym raportem?

18. Poświęć trochę czasu na ulepszenie formatowania raportu, aby każdy, kto czyta wynik, mógł zrozumieć jego znaczenie.

19. Spotkaj się ze swoją grupą, aby omówić ulepszenia, które należy wprowadzić do skryptu. Wyjaśnij te ulepszenia. Jakie inne informacje mogłyby być przydatne dla specjalisty ds. bezpieczeństwa w tym raporcie? 20. Wybierz rzecznika ze swojej grupy, który wygłosi trzy- lub pięciominutową prezentację na temat ostatecznego scenariusza i wyjaśni, dlaczego Twój program jest najbardziej rynkowy. Po wszystkich prezentacjach niech klasa wybierze zwycięzcę.

21. Zamknij wszystkie otwarte okna.

ZROZUMIENIE PYTHONA

Jak powiedział kiedyś inny znany Python: „A teraz coś zupełnie innego”. Język programowania Python został nazwany na cześć programu telewizyjnego BBC, Monty Python’s Flying Circus. Python, podobnie jak Perl, jest językiem skryptowym z pewnymi obiektowymi cechami. Jako język skryptowy jest popularny do szybkiego tworzenia małych i średnich programów. Języków skryptowych nie trzeba kompilować, ponieważ są interpretowane, co prawdopodobnie sprawia, że są szybsze podczas tworzenia, testowania i wykonywania krótkich programów. Etyczni hakerzy często tworzą programy, które pomagają zautomatyzować ich działania, a Python doskonale nadaje się do tego celu. W rzeczywistości Python zyskuje na popularności w społeczności etycznego hakowania. Biblioteki hakerskie i narzędzia napisane w Pythonie stają się szeroko dostępne. Python kładzie nacisk na

czytelność kodu i używa wcięć do definiowania bloków kodu (a nie nawiasów i klamer, jak w Perl lub C). Użycie nieprawidłowego odstępu może spowodować błąd składniowy w Pythonie, więc pamiętaj, aby wcięcia instrukcji były poprawne i spójne. W tej sekcji zapoznasz się ze składnią języka Python i przećwiczysz pisanie skryptów w Pythonie.

Tło języka Python

Guido van Rossum wymyślił Pythona pod koniec lat 80. i rozpoczął jego implementację w grudniu 1989 r. Jest głównym autorem Pythona i wciąż odgrywa centralną rolę w podejmowaniu decyzji dotyczących kierunku rozwoju Pythona. Python może działać na niemal każdej platformie (w tym Windows), a systemy operacyjne oparte na *nix-ach zazwyczaj mają już zainstalowanego Pythona. Python 3.9 (wydany w październiku 2020 r.) to niedawna stabilna wersja. Aby uzyskać więcej informacji, odwiedź https://en.wikipedia.org/wiki/Python_(programming_language).

Pisanie skryptu Pythona przy użyciu GVim

Czas trwania: 60 minut

Cel: Napisz skrypt Pythona przy użyciu GVim.

Opis: Specjaliści ds. bezpieczeństwa i hakerzy coraz częściej używają Pythona. Coraz więcej programów hakerskich jest pisanych w Pythonie, więc wszelkie umiejętności, które rozwiniesz w tym języku, pomogą Ci w karierze. W tej aktywności napiszesz podstawowy skrypt Pythona.

1. Uruchom komputer w systemie Kali Linux, otwórz okno terminala, a następnie zmień katalog na pulpit za pomocą polecenia cd ~/Desktop.

2. Wpisz gvim first.py i naciśnij Enter.

3. Wybierz kartę Składnia u góry okna GVim, a następnie wybierz opcję Automatyczne, aby włączyć podświetlanie składni dla projektu Python.

4. Naciśnij klawisz i, aby przejść do trybu wstawiania.

5. W pierwszym wierszu wpisz # this is my first Python script program i naciśnij Enter, aby dodać jeden z trzech początkowych komentarzy.

6. Wpisz # I should always have comments in my scripts! i naciśnij Enter dwa razy.

7. Wpisz # ten kod wyświetli na ekranie „Hello security testers!” i naciśnij Enter.

8. Wpisz print(‘Hello security testers!’) i naciśnij Enter, aby program wyświetlił określony tekst.

9. Zapisz plik.

10. W oknie terminala wpisz python first.py i naciśnij Enter, aby uruchomić program. Jeśli nie popełniłeś żadnych błędów. Jeśli otrzymałeś komunikaty o błędach, przeczytaj swój plik i porównaj go z wierszami kodu w krokach tej aktywności. Jeśli poprawisz jakiekolwiek błędy, zapisz plik i powtórz krok 10

Zrozumienie podstaw języka Python

Przydatna jest wiedza, jak szybko uzyskać pomoc w dowolnym języku programowania. Polecenie python -h wyświetla parametry używane z poleceniem python. Witryna https://python.org/ jest doskonałym repozytorium informacji o języku Python. Znajdują się na niej samouczki i często zadawane pytania (FAQ), do których możesz uzyskać dostęp, aby dowiedzieć się więcej.

Zrozumienie BLT języka Python

Jak się wcześniej dowiedziałeś, wszystkie języki programowania muszą mieć sposób na rozgałęzianie, pętlenie i testowanie. Poniższe sekcje wykorzystują przykłady kodu, aby pokazać, jak Python obsługuje te funkcje BLT. Podczas analizowania przykładów pamiętaj o następujących zasadach składni:

• Odstępy są ważne. Python używa odstępów do określania bloków kodu. Perl używa nawiasów klamrowych do definiowania bloków kodu w instrukcjach if i funkcjach; Python tego nie robi. Możesz mieć błąd składniowy, nie wcinając poprawnie wierszy kodu.

• Podczas tworzenia funkcji wstaw słowo kluczowe def przed nazwą funkcji.

• Zmienne nie zaczynają się od żadnego symbolu specjalnego. Perl używa symbolu $ do wskazania zmiennej, ale Python nie.

• Na końcu wierszy kodu nie ma znaków specjalnych. Perl używa znaku średnika (;), ale Python nie.

• Wiersze komentarzy zaczynają się od symbolu #.

Rozgałęzienie w Pythonie

Aby przejść z jednej funkcji do drugiej w programie Python, wywołujesz funkcję, wpisując jej nazwę, a następnie nawiasy. W Pythonie funkcja musi być zdefiniowana, zanim będzie można ją wywołać; w Perlu nie miało to znaczenia. Na przykład funkcja name_best_guitarist() musi zostać wstawiona do kodu, zanim będzie można ją wywołać. W poniższym programie Python linia name_best_guitarist() rozgałęzia program do funkcji name_best_guitarist():

```
# Python program illustrating the branching function

# Documentation is important

# Initialize variables

first_name = "Jimi "

last_name = "Hendrix"

# define the name_best_guitarist function

# a function must be defined before it can be called

def name_best_guitarist():

print(first_name + last_name + " was the best!")

name_best_guitarist()
```

Wiele funkcji przyjmuje parametry. Na przykład funkcja add_these_numbers() przyjmuje dwa parametry jako dane wejściowe, liczby do dodania:

```
def add_these_numbers(x,y):

z=x+y

print(z)
```

Funkcje, które nie oczekują parametrów, są wywoływane bez niczego w nawiasach, jak w przypadku wywołania name_best_guitarist() w poprzednim przykładzie.

Pętle w Pythonie

Python ma pętle for i while, tak jak Perl i C, chociaż składnia jest inna. W tej sekcji poznasz te dwa mechanizmy pętli.

Pętla for w Pythonie powtarza się, dopóki nie przejdzie przez każdy element określony na liście elementów. Nie jest to pętla licznika przyrostowego, taka jak w C i Perlu. Na przykład, jeśli masz listę nazw i chcesz użyć pętli for, aby wydrukować każdą nazwę, możesz napisać następujący kod Pythona:

```
names = ["Bob", "Jamal", "Sasha"]

for x in names:

print(x)
```

Jeśli chcesz utworzyć pętlę for liczącą od liczby początkowej do liczby końcowej, możesz użyć listy liczb lub funkcji o nazwie range():

```
for x in range(6):

print(x)
```

W poprzednim przykładzie wywołanie funkcji range(6) tworzy sekwencję liczb zaczynającą się od 0 i kończącą na 5, w sumie sześć liczb. Jeśli chcesz liczyć od 1 do 6, musisz zmodyfikować kod, aby print(x+1) zamiast print(x). Pamiętaj o tym, gdy używasz funkcji range() w swoim programowaniu. Pętla while w Pythonie jest podobna do pętli while w Perlu i C. Pętla while powtarza zestaw wierszy kodu tak długo, jak warunek testowy pozostaje prawdziwy. Pamiętaj, że Python używa wcięć do oznaczania bloków kodu, więc nie potrzebujesz nawiasów w pętli while. W poniższym przykładzie pętla while będzie kontynuować pętlę (drukując wartość zmiennej i, a następnie zwiększając zmienną i o 1), dopóki zmienna i jest mniejsza od 6.

```
i = 1

while i < 6:

print(i)

i += 1
```

Testowanie warunków w Pythonie

Python ma operatory do porównań (testy logiczne) i obliczeń matematycznych. Standardowe operatory Pythona są identyczne z operatorami Perla. Tabela 7-11 zawiera listę operatorów, których można używać w Pythonie.

Przykład funkcji operatora

+ : Dodawanie : suma = sal + prowizja

– : Odejmowanie : zysk = sprzedaż brutto – kosztTowarów

* : Mnożenie : suma = koszt * ilość

/ : Dzielenie : GPA = sumaPunktów / liczbaKlas

% : Moduł : x = a % 2

** : Wykładnik : powierzchnia = 3,14 * (r**2)

Przypisania

= : Przypisanie : nazwisko = "Rivera"

+= : Dodaj, a następnie przypisanie : a+ = 10 #skrócona forma dla a=a+10

-= : Odejmij, a następnie przypisanie : a–=10 #skrócona forma dla a=a–10

= : Pomnóż, a następnie : a = 10 #skrócona forma dla a=a*10

/= Podziel, a następnie przypisanie a/ = 10 #skrócona forma dla a=a/10

%= : Moduł, następnie przypisanie : a%=10 #skrót dla a=a%10

= : Wykładnik i przypisanie : a=2 #skrót dla a=a**2

++ : Inkrementacja : a++ #zwiększenie a o 1

— : Dekrementacja : a— #zmniejszenie a o 1

Porównania

== : Równe : a== 1 #porównanie wartości a z 1

!= : Nierówne : a!=1 #a nie jest równe 1

> : Większe niż: a>10

< : Mniejsze niż : a<10

>= : Większe lub równe : a>=10

<= : Mniejsze lub równe : a<=10

Instrukcje if i operatory logiczne

Operatory porównania wymienione w tabeli 7-11 są również znane jako operatory logiczne. Operatorów logicznych używa się w instrukcjach if. Instrukcja if łączy operatory logiczne ze zmiennymi i liczbami, aby tworzyć kontrole warunkowe. Użyj instrukcji if, aby Twój kod wykonywał określone operacje, jeśli logiczna kontrola warunkowa jest prawdziwa, a być może wykonywał inną operację, jeśli nie jest prawdziwa. Możesz łączyć instrukcję if ze słowami kluczowymi else i elseif oraz zagnieżdżać instrukcje if, aby tworzyć złożone kontrole logiczne. Poniżej przedstawiono kilka przykładów if, else, elif, zagnieżdżonych if i kontroli warunków używanych do kontrolowania, jaki kod jest wykonywany.

• if — sprawdza, czy warunek jest prawdziwy. Przykład:

if (age < 12)

print("Musisz być wszechwiedzący!")

• else — używane, gdy istnieje tylko jedna opcja do wykonania, jeśli warunek nie jest prawdziwy. Przykład:

if (age > 12)

print("Musisz być wszechwiedzący!")

else

print("Przepraszam, ale nie wiem, dlaczego niebo jest niebieskie.")

• elif — używane, gdy istnieje kilka warunków do przetestowania. Przykład:

```
if ( (age > 12) && (age < 20) )

print("Musisz być wszechwiedzący!")

elif (age > 39)

print("Musisz skłamać o swoim wieku!")

else:

print("Aby być młodym...")
```

• Możesz również umieszczać instrukcje if wewnątrz innych instrukcji if. Są one nazywane zagnieżdżonymi if. Przykład:

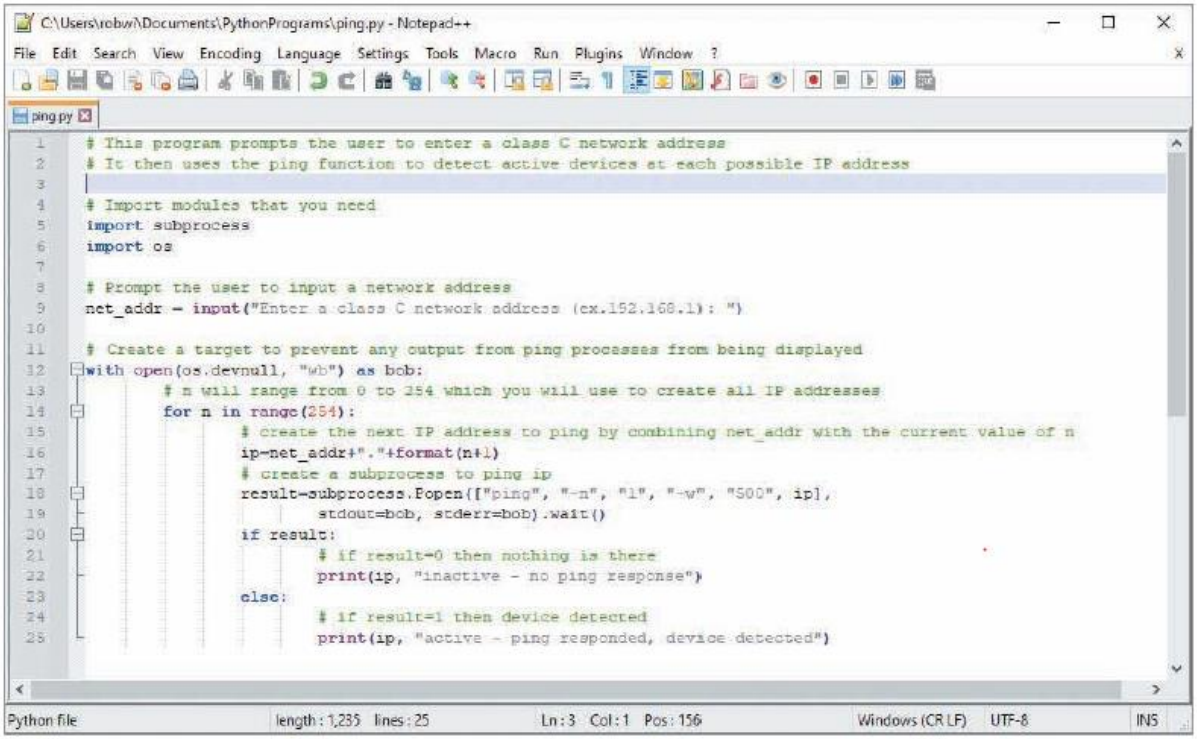
```
y = 69

if y > 10:

print("Większe niż dziesięć")

if y > 20:

print("Również większe niż 20!")

else:

print("Ale nie większe niż 20")
```

Aktywność 7-8: Pisanie skryptu Pythona do testowania bezpieczeństwa

Czas trwania: 30 minut

Cel: Napisz skrypt Pythona, który używa komponentów rozgałęziania, pętli i testowania.

Opis: Specjaliści ds. bezpieczeństwa często muszą automatyzować lub tworzyć narzędzia, które pomogą im przeprowadzać testy bezpieczeństwa. W tej aktywności napiszesz skrypt Pythona, który używa polecenia ping i pętli for, aby pingować numery IP dla całej sieci klasy C. Twój instruktor poda Ci adres sieci klasy C. Jeśli nie jesteś na zajęciach, wybierz sieć klasy C według własnego wyboru (być może sieć domową). Jeśli ping zakończy się powodzeniem, oznacza to, że pod tym adresem IP znaleziono urządzenie komputerowe. Zamierzasz pobrać i zainstalować najnowszą wersję Pythona dla systemu Windows i wykonać tę aktywność w środowisku Windows. Pamiętaj, że nieprawidłowe wcięcia mogą powodować błędy składniowe w Pythonie, więc upewnij się, że Twoje wcięcia są zgodne z tym, co pokazano na rysunku.



1. Otwórz przeglądarkę internetową i przejdź do www.python.org/downloads/. Pobierz i zainstaluj najnowszą wersję Pythona dla systemu Windows. Upewnij się, że zaznaczyłeś pole wyboru, aby dodać Pythona do zmiennej PATH.

2. Zapisz adres IP, który ma być używany dla sieci klasy C.

3. Uruchom Notepad++ i utwórz nowy plik o nazwie ping.py.

4. Wpisz # Ten program prosi użytkownika o podanie adresu sieci klasy C i naciśnięcie Enter.

5. Wpisz #, a następnie używa funkcji ping do wykrywania aktywnych urządzeń na każdym możliwym adresie IP i naciśnij Enter dwa razy.

6. Wpisz # Importuj potrzebne moduły i naciśnij Enter.

7. Wpisz import subprocess i naciśnij Enter.

8. Wpisz import os i naciśnij Enter dwa razy.

9. Wpisz # Monituj użytkownika o podanie adresu sieciowego i naciśnij Enter.

10. Wpisz net_addr = input("wprowadź adres sieci klasy c (ex.192.168.1): ") i naciśnij Enter dwa razy.

11. Wpisz # Utwórz cel, aby zapobiec wyświetlaniu wyników procesów ping i naciśnij Enter.

12. Wpisz z open(os.devnull, "wb") jako bob: i naciśnij Enter.

13. Naciśnij klawisz Tab raz, aby wciąć, wpisz # n będzie miało zakres od 0 do 254, którego użyjesz do utworzenia wszystkich adresów IP, a następnie naciśnij Enter.

14. Naciśnij klawisz Tab raz, aby wciąć, a następnie wpisz for n in range(254): i naciśnij Enter.

15. Naciśnij klawisz Tab dwa razy, a następnie wpisz # utwórz następny adres IP do pingowania, łącząc net_addr z bieżącą wartością n i naciśnij Enter.

16. Naciśnij klawisz Tab dwa razy, a następnie wpisz ip=net_addr+"."+format(n+1) i naciśnij Enter.

17. Naciśnij klawisz Tab dwa razy, a następnie wpisz # utwórz podproces do pingowania IP i naciśnij Enter.

18. Naciśnij klawisz Tab dwa razy, a następnie wpisz result=subprocess.Popen(["ping", "-n", "1", "-w", "500", ip], stdout=bob,

stderr=bob).wait() i naciśnij Enter.

19. Naciśnij klawisz Tab dwa razy, a następnie wpisz if result: i naciśnij Enter.

20. Naciśnij klawisz Tab trzy razy, a następnie wpisz # if result=0 then nothing is i naciśnij Enter.

21. Naciśnij klawisz Tab trzy razy, a następnie wpisz print(ip, "inactive - no ping response") i naciśnij Enter.

22. Naciśnij klawisz Tab dwa razy, a następnie wpisz else: i naciśnij Enter.

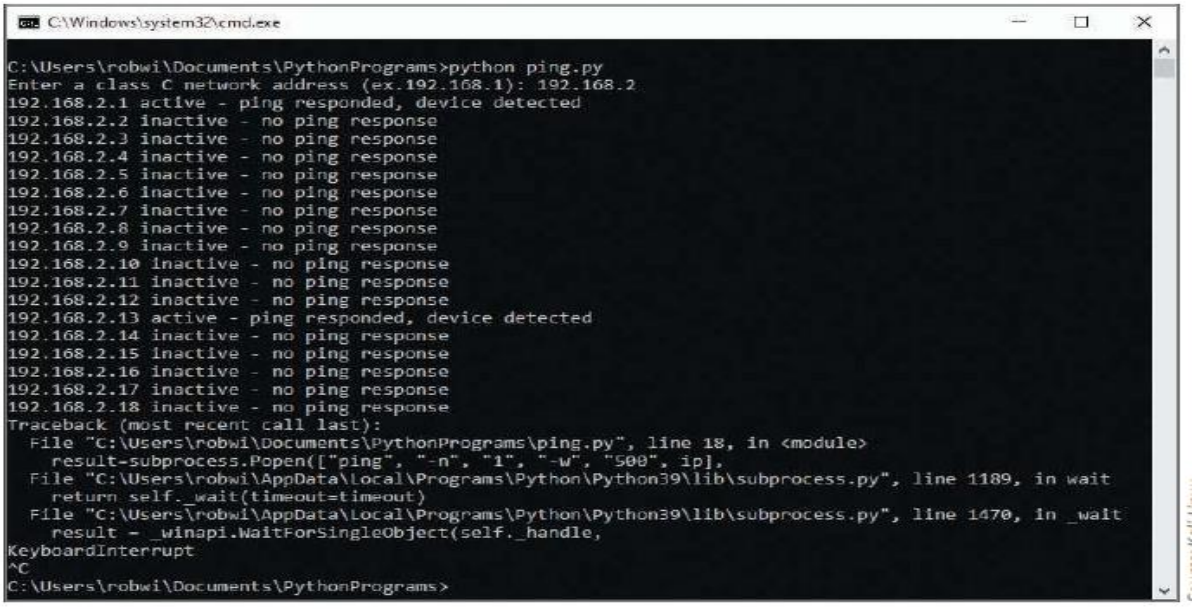
23. Naciśnij klawisz Tab trzy razy, a następnie wpisz # if result=1 then device detected i naciśnij Enter.

24. Naciśnij klawisz Tab trzy razy, a następnie wpisz print(ip, "active - ping answered, device detected") i naciśnij Enter.

25. Przejdź przez każdy wiersz kodu i upewnij się, że składnia jest poprawna. Zobacz Rysunek 7-19. Zwróć uwagę, że w wierszu 18 jednym z parametrów, których używa polecenie ping, jest liczba jeden („1”), a nie litera „l”. Upewnij się również, że wcięcie jest poprawne. Notepad++ automatycznie wtnie niektóre sekcje, ale upewnij się, że wcięcie jest zgodne z wcięciem pokazanym na Rysunku.

26. Zapisz plik w wybranej lokalizacji. Będziesz musiał przejść do pliku z wiersza poleceń systemu Windows, więc zapisz go w łatwej do zapamiętania lokalizacji.

27. Otwórz wiersz poleceń systemu Windows i przejdź do folderu zawierającego plik ping.py. Uruchom skrypt, wpisując python ping.py i naciskając Enter. Jeśli nie ma błędów, program powinien rozpocząć pingowanie adresów IP, jak pokazano na Rysunku .



28. Możesz pozwolić programowi działać do końca lub, aby zakończyć skrypt Pythona, nacisnąć ctrl+c.

Python Shell (REPL)

Python ma również interaktywną powłokę, w której można wprowadzać polecenia Pythona i natychmiast je wykonywać. Ta powłoka jest również znana jako REPL, co oznacza Read, Evaluate, Print, Loop. REPL odczytuje polecenie, ocenia polecenie, drukuje wyniki i zapętla się, aby odczytać więcej poleceń. Ta powłoka jest wygodna do wykonywania szybkich zadań, takich jak obliczenia lub wykonywanie operacji przez wywoływanie istniejących funkcji. Możesz wejść do powłoki, wpisując python i naciskając Enter w terminalu lub oknie poleceń.

Programowanie obiektowe w Pythonie

Python zawiera OOP, a także model programowania funkcjonalnego, którego używałeś do tej pory. Python obsługuje tradycyjne koncepcje OOP, takie jak klasy, obiekty i dziedziczenie. Chociaż badanie OOP wykracza poza zakres tej książki, możesz zacząć uczyć się więcej o używaniu OOP w Pythonie, korzystając z samouczka pod adresem https://docs.python.org/3/tutorial/classes.html. Program definiuje klasę o nazwie Dog i używa jej do tworzenia instancji i przypisywania atrybutów trzem obiektom Dog: dog1, dog2 i dog3. Następnie wywołuje funkcję członkowską klasy Dog showInfo(), aby wyświetlić atrybuty obiektów Dog.

PRZEGLĄD RUBY

Innym obiektowym językiem używanym przez wielu testerów bezpieczeństwa jest Ruby, który jest podobny do Perla. Testerzy bezpieczeństwa używają również Metasploit (www.metasploit.com), programu opartego na Ruby, aby sprawdzać luki w zabezpieczeniach systemów komputerowych. Metasploit zawiera setki exploitów, które można uruchomić na komputerze lub w sieci ofiary, co czyni go przydatnym narzędziem dla hakerów. Testerzy bezpieczeństwa używający Metasploit powinni rozumieć podstawy Ruby i umieć modyfikować kod Ruby, aby dostosować go do różnych środowisk i celów. Na przykład testerzy bezpieczeństwa mogą musieć zmodyfikować kod modułu odwrotnej powłoki w Ruby, aby był zgodny z systemem docelowym, w którym przeprowadzają testy podatności (patrz Rysunek 7-23). Odwrotna powłoka to tylne wejście inicjowane z sieci docelowej, które umożliwia przejęcie kontroli nad celem nawet wtedy, gdy znajduje się on za zaporą sieciową. Wyszukaj w Internecie „Reverse Shell”, aby dowiedzieć się więcej na ten temat. Rysunek 7-24 pokazuje niektóre z wielu exploitów napisanych w Ruby. Zwróć uwagę na rozszerzenie .rb dla Ruby w nazwach programów. Tester bezpieczeństwa otworzył moduł dla exploita podatności MS15-020 w vim do edycji. Jak widać, składnia Ruby jest podobna do składni programowania obiektowego, a moduł zawiera szczegółowe opisy kodu Ruby.

PODSUMOWANIE MODUŁU

• Pisanie algorytmu i używanie pseudokodu to dobre nawyki, które należy przyjąć podczas pisania programów.

• Przejrzysta dokumentacja kodu programu jest niezbędna.

• C, Perl i Python to popularne języki programowania zarówno dla profesjonalistów ds. bezpieczeństwa, jak i hakerów.

• Poznanie BLT dowolnego języka programowania może pomóc w opanowaniu podstaw programowania. Rozgałęzianie, pętlenie i testowanie to najważniejsze aspekty programowania

• Dostępnych jest wiele kompilatorów C. GNU GCC to kompilator C typu open source dołączony do większości implementacji Linuksa.

• HTML to podstawowy język używany do tworzenia stron internetowych. Profesjonaliści ds. bezpieczeństwa muszą rozpoznawać, kiedy coś wygląda podejrzanie na stronie internetowej, więc powinni być w stanie odczytać plik HTML.

• Profesjonaliści ds. bezpieczeństwa powinni mieć podstawową wiedzę na temat Perla, Pythona i C, ponieważ wiele narzędzi bezpieczeństwa jest napisanych w tych językach. Profesjonaliści ds. bezpieczeństwa, którzy rozumieją te języki programowania, mogą modyfikować narzędzia bezpieczeństwa i tworzyć własne, dostosowane narzędzia.

• Dzięki programowaniu obiektowemu programiści mogą tworzyć klasy, które są strukturami zawierającymi zarówno dane, jak i funkcje. Funkcje w tych klasach to programy wykonujące określone zadania.

• WinAPI (dawniej nazywany Win32 API) to interfejs do systemu operacyjnego Windows, którego programiści mogą używać do uzyskiwania dostępu do informacji o komputerze z systemem Windows, takich jak nazwa komputera, nazwa systemu operacyjnego itd.

• Python to język skryptowy, który obsługuje zarówno staroszkolny paradygmat funkcjonalny, jak i obiektowy model programowania. Python używa wcięć do oznaczania bloków kodu, a nie nawiasów klamrowych, jak w przypadku języków C i Perl.

• Ruby to elastyczny, obiektowy język programowania podobny do Perla. Testerzy bezpieczeństwa i atakujący używają Metasploit, zawierającego moduły exploitów napisane w języku Ruby, do sprawdzania luk w zabezpieczeniach lub atakowania systemów