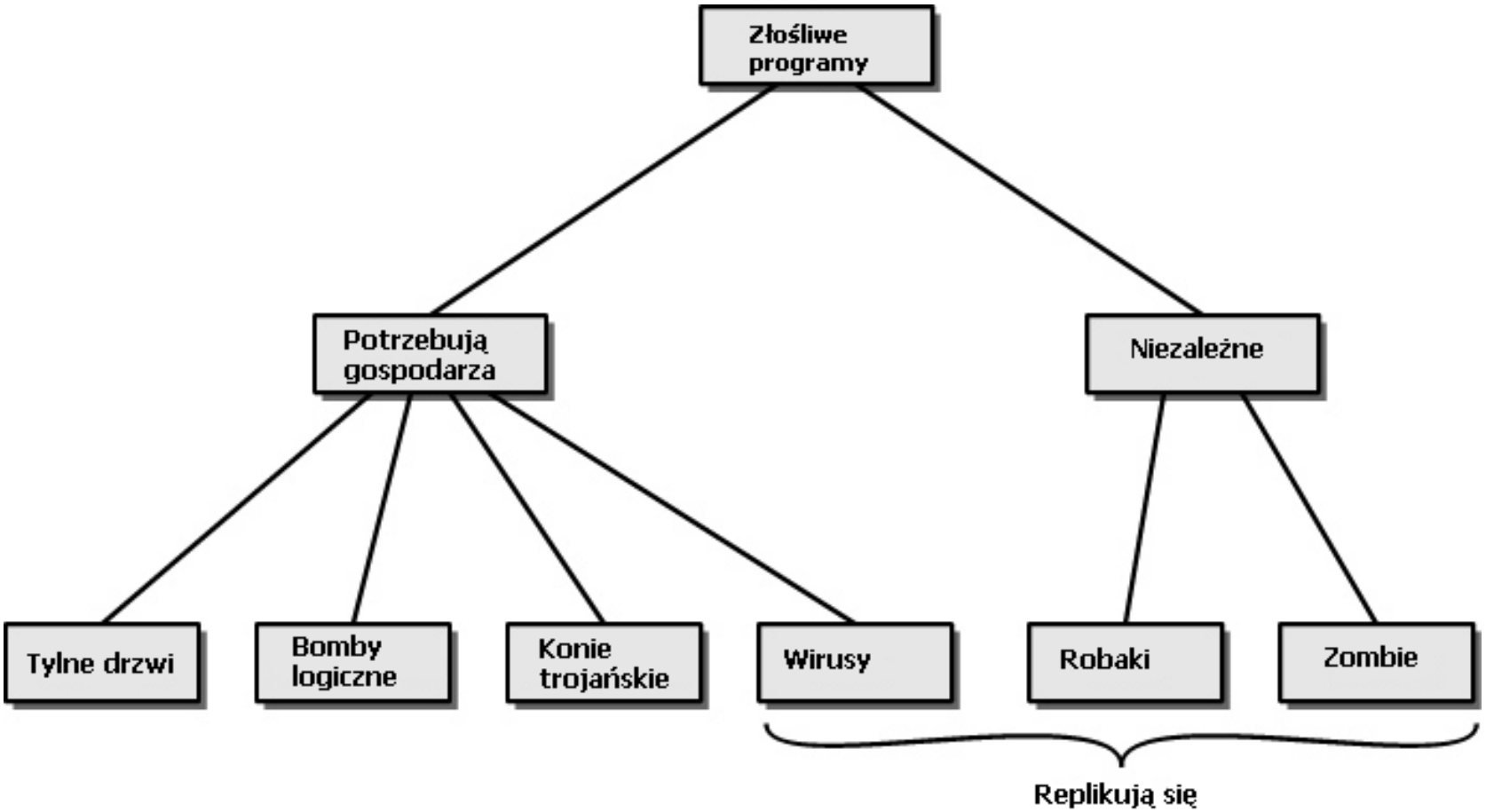
Wykład 12

Bezpieczeństwo c.d.

Złośliwe programy - klasyfikacja



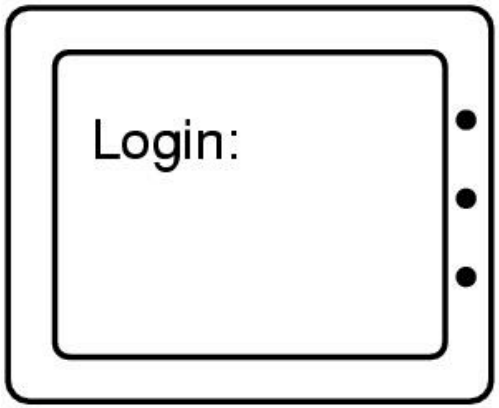
Tylne drzwi + bomby logiczne

- ***Tylne drzwi (ang. trapdoor)*** Mechanizm, pozwalający osobie która jest świadoma jego istnienia na uzyskanie dostępu.
- Przykład: każdy który wykona telnet na port 12345 otrzymuje uprawnienia Administratora.
- Programiści często używają tylnych drzwi w celu ułatwienia debugowania systemu (mogą zapomnieć je usunąć, przykład jednej z bazy danych, tylne drzwi wykryto dopiero, gdy firma udostępniła kod na zasadach Open Source).
- Włamywacze pozostawiają/instalują tylne drzwi aby ułatwić sobie (i być może innym kolejne włamania).
- ***Bomba logiczna***, kod umieszczony w legalnym programie, który eksploduje gdy spełnione są pewne warunki np.
  - Programista zostanie skreślony z listy płac firmy.
  - Firma pisząca program nie otrzyma zapłaty.

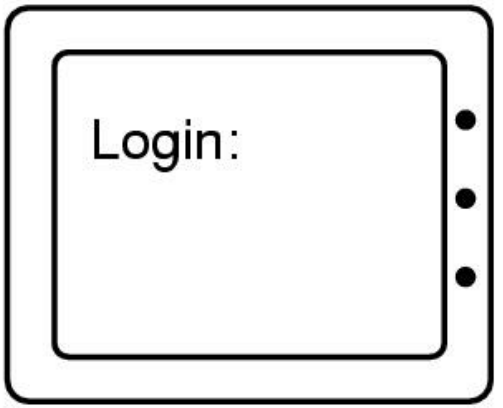
Konie trojańskie

- Program, często atrakcyjny (np. gra komputerowa, program kompresji plików, ...), często dostępny publicznie w internecie, który dodatkowo wykonuje pewne złośliwe czynności.
  - Np. co 12 miesięcy formatuje dysk.
  - Także programy typu spyware.

- Przykład: login spoofing:



(a)



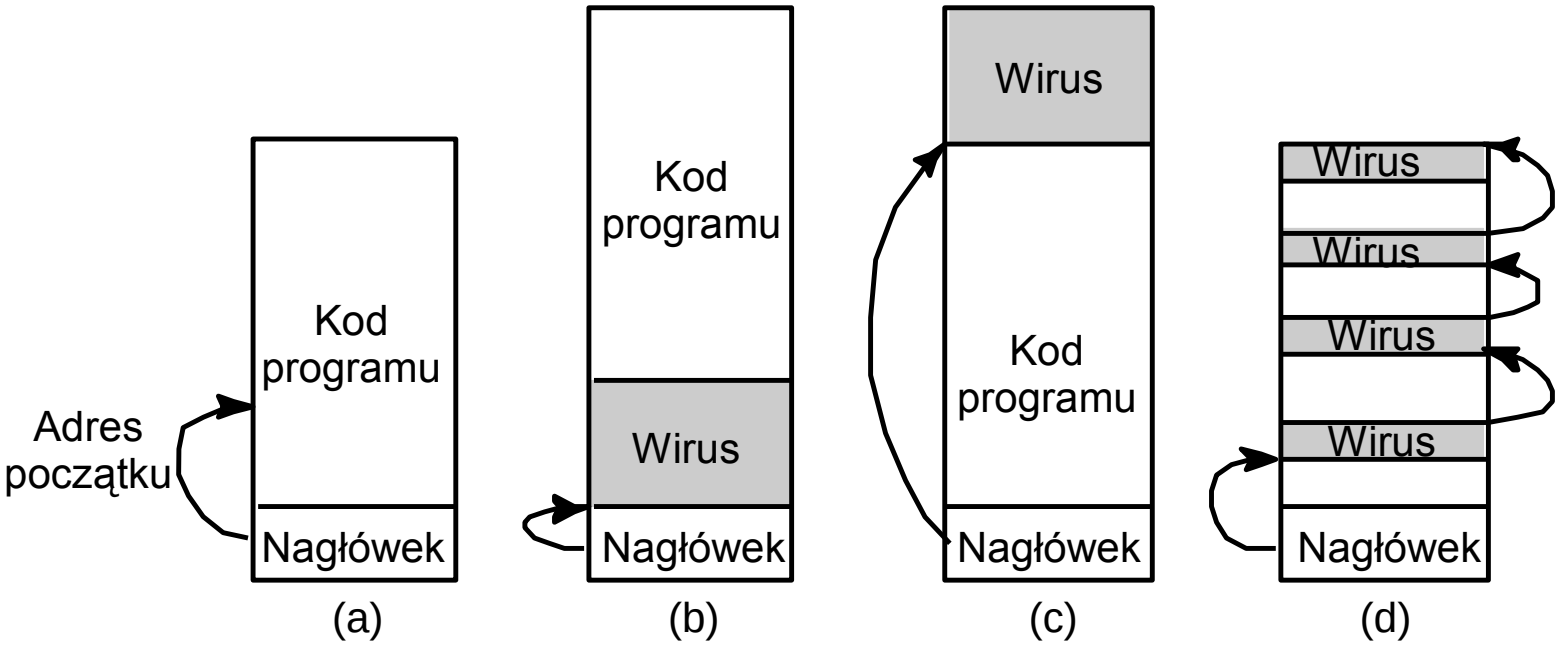
(b)

(a) Fałszywy ekran logowania

(b) prawdziwy ekran logowania

:

Wirusy



- Wirus - program zdolny do powielania przez dodanie swojego kodu do innego programu. Uruchomienie innego programu powoduje ponowne powielenie się wirusa.
- Wirusy mogą także infekować: sektor startowy (boot sektor), dokumenty programów aplikacyjnych np. pliki Worda i Excela (bo mogą zawierać kod Visual Basica uruchamiany automatycznie po wczytaniu dokumentu - dziękujemy ci M$) a nawet kod źródłowy (np. w języku C)

Ochrona przed wirusami

- Generalnie nie należy uruchamiać niepewnego kodu, oprócz tego specjalistyczne oprogramowanie typu:
- Skanery antywirusowe: porównują kod wirusa z bazą danych.
  - Wirusy starają się unikać wykrycia: szyfrowanie, wirusy polimorficzne.
  - Skanowanie trwa długo.
  - Jeżeli wirusa nie ma w bazie danych - problemy.
- Sprawdzanie spójności: utrzymuj sumy kontrolne dla każdego pliku (wirus może próbować je zmienić)
- Blokery: Uniemożliwiaj podejrzane zachowanie np. zapis do plików .exe - ale robi to wiele legalnych programów.
- Kontrola zachowania (ang. behavioral checking) wykrywaj zmiany zachowania się programu - przedmiot badań naukowych.

Wykrywanie włamań (ang. intrusion detection)

- Skanery antywirusowe są przykładem szerszej klasy programów służących do modyfikacji włamań do systemu. Programy te możemy podzielić na następujące grupy:
- Wykrywanie sygnatur (np. monitoruj pliki szukając sygnatur wirusów, monitoruj pakiety sieciowe w poszukiwaniu sygnatur exploitów)
- Wykrywanie anomalii: np. monitoruj sumy kontrolne plików .exe i niektórych plików systemowych (np. /etc/passwd), alarmuj gdy logi systemowe zmniejszyły rozmiar lub pakiet przesłany przez sieć zawiera tekst "/etc/passwd"
  - Szczególnym przypadkiem wykrywanie anomalii jest monitorowanie sekwencji wywołań systemowych dokonywanych przez program wykonujący się z przywilejami administratora.
  - Np. system nauczył się że program wywołuje wywołania systemowe w następującej kolejności: open,read,mmap,mmap,getrlimit,close.
  - Zmiana tej sekwencji powoduje wszczęcie alarmu.
- Bardzo często w.w programy zbierają informacje z logów systemowych (np. katalog /var/log w Uniksach)
- Problemy: (takie same jak w przypadku antywirusów): wykrywanie sygnatur nie reaguje na nieznane metody włamania, a wykrywanie anomalii może prowadzić do fałszywych alarmów.

Ataki typu Denial of Service

- Przykład: w nagłówku pakietu w jednym z protokołów TCP/IP pewne pole przechowuje długość pakietu.
  - System Windows 95 nie sprawdzał, czy te pole ma sensowną wartość. (Tzn. zakładał że wartość ta jest zawsze poprawna).
  - W efekcie wystarczyło wysłać przez sieć pakiet o wartości pola długość = 0xffffffff. System Windows przyjmował to za pakiet o długości 4GB i usiłując go "skopiować" zamazywał cały RAM komputera.
- Inny przykład: dzięki błędowi w procesorach Intela każdy użytkownik mógł wykonać niedozwoloną instrukcję o prefiksie 0x0f i "wyłączyć" procesor -> na szczęście odnaleziono obejście programowe.
- Nowość: ataki typu DDOS (Distributed DOS). Włamywacze przejmują kilka (set,tysięcy) komputerów w sieci (zombi). Następnie z każdego komputera *częściowo* otwierają setki połączeń TCP np. do serwera WWW - co przeciąża serwer, tak że zwykli użytkownicy nie mogą z niego korzystać.
  - Obrona przed takimi atakami jest bardzo trudna, ponieważ ciężko stwierdzić że w ogóle trwa atak (może serwer jest obciążony legalnym ruchem)

Robaki (ang. Worms)

- Używa połączenie sieciowego do przenoszenie się z jednego systemu na drugi. Nie wymaga nosiciela.
- Robaki mogą się przenosić przy pomocy:
  - E-maili. Otwarcie e-maila automatycznie uruchamia załącznik (dziękujemy ci M$). Załącznik zawiera program, który przegląda książkę adresową i rozsyła się do wszystkich korespondentów.
  - Przydaje się odrobina inżynierii społecznej np. tytuł maila “I Love You”.
- Możliwości zdalnego wykonania kodu - dzięki wykorzystaniu techniki przepełnienia bufora.
  - Błąd w systemie operacyjnym (często nie w jądrze, ale w uprzywilejowanych programach dokonujących komunikacji sieciowej użytkownika typu serwer WWW, serwer poczty) pozwala atakującemu na ***zdalne*** wykonanie dowolnego kodu.
  - Ten kod transmituje robaka z maszyny atakującego, uruchamia robaka, skanuje sieć w poszukiwaniu innych ofiar itp.

Przepełnienie bufora (1)

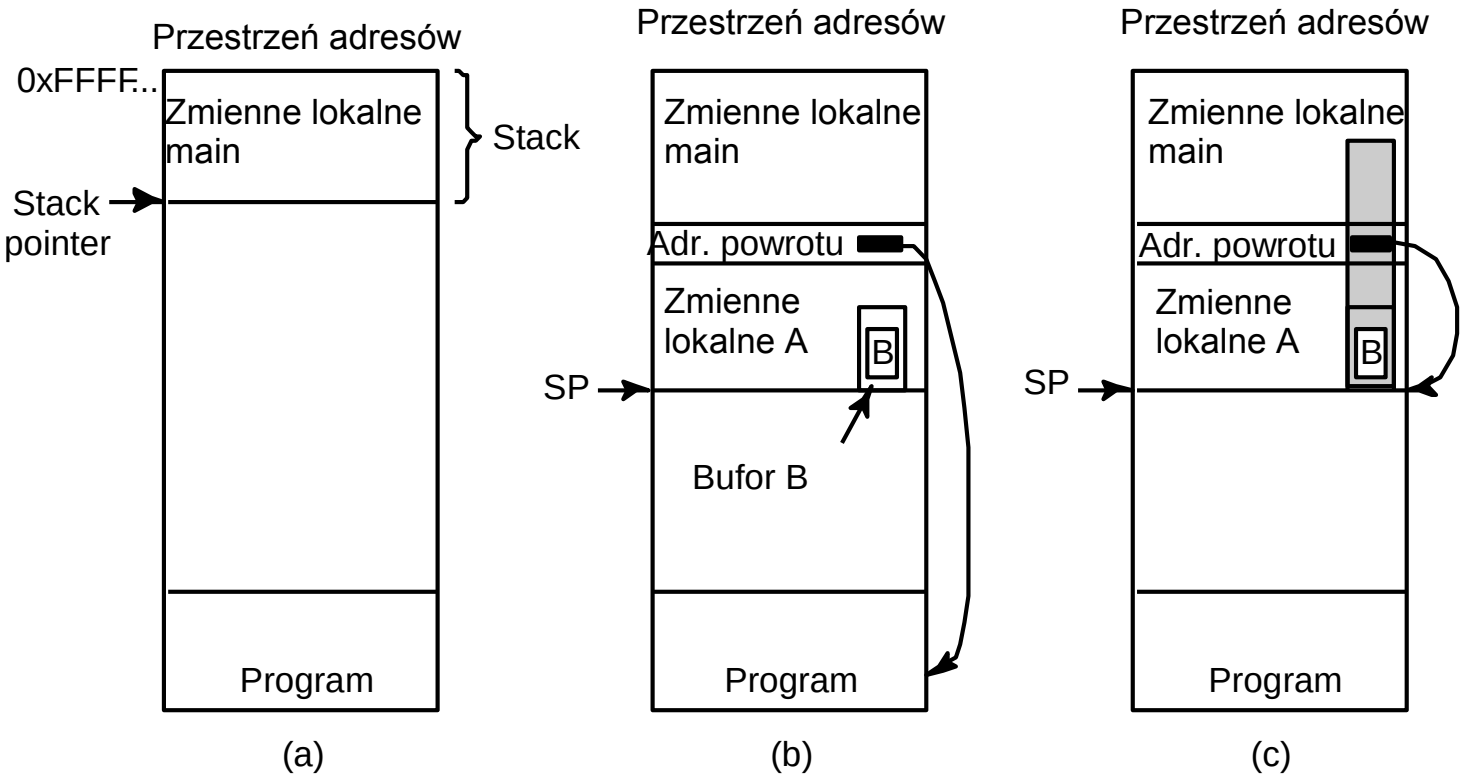
- Wyobraźmy sobie program wykonujący sie wysokimi uprawnieniami (np. Administratora - patrz bit suid) a w w funkcja A wołana z funkcji main:

```
void A() {
   char B[128]
   scanf("%s",Bufor);


   // dalszy kod funkcji A
}
```

- Generalnie chodzi o sytuację w której (a) program otrzymuje dane z zewnątrz(b) na dane zaalokowaliśmy bufor o stałej długości, (c) nie sprawdzamy, czy dane przesyłane z zewnątrz nie zajmują więcej miejsca niż na nie przeznaczyliśmy.
- Co się stanie jeżeli napis będzie dłuższy od 128 znaków ???

Przepełnienie bufora (2)



- Zmienne lokalne oraz adresy powrotu z funkcji (instr. call) przechowywane są na stosie.
- W wyniku przepełnienia bufora adres powrotu zostaje zamazany. Przy złośliwym skonstruowaniu danych, adres powrotu może wskazać na inną część bufora, co daje atakującemu możliwość wykonania dowolnego kodu z uprawnieniami zaatakowanego procesu (np. wyświetlenie shella)

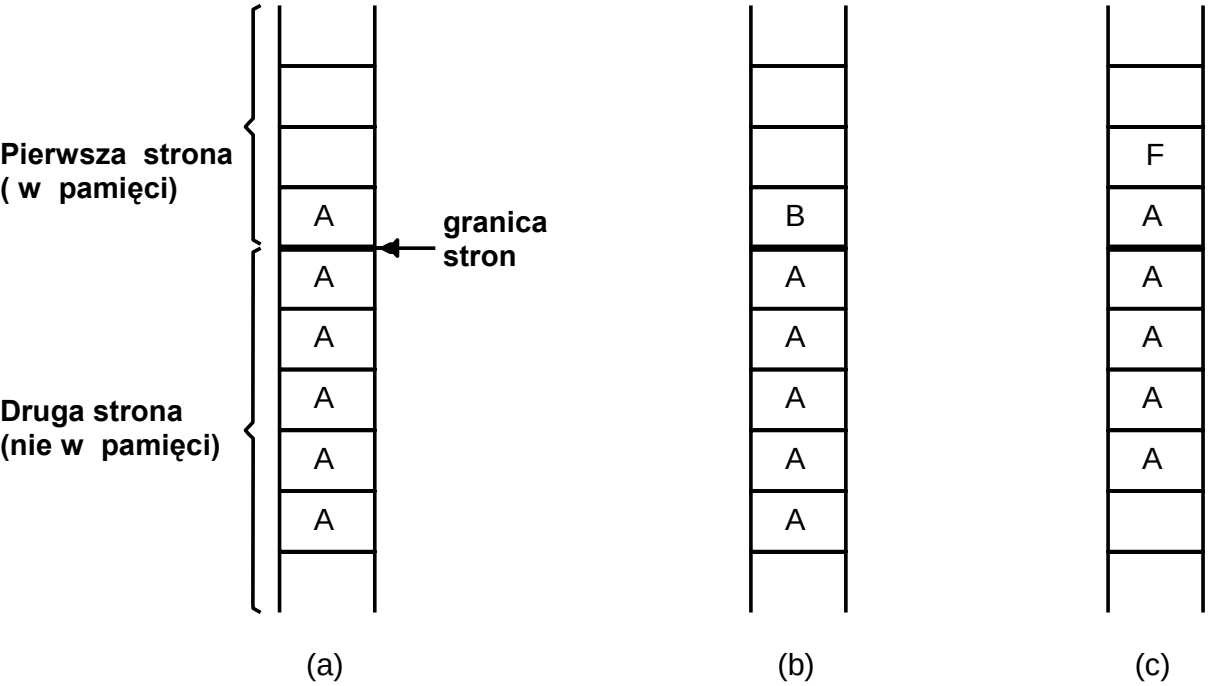
Jak walczyć z przepełnieniami bufora

- Generalnie nie wolno ufać danym przekazanym na wejściu procesu (dotyczy to zwłaszcza procesów o wysokich uprawnieniach).
- Do przetwarzania danych używać wyłącznie funkcji pozwalających na określenie rozmiaru bufora (strcpy => strncpy).
- Ale mimo to firma M$ miała olbrzymie problemy z przepełnieniami bufora (robaki sieciowe NIMDA, CODE RED).
  - W sieci szybko pojawiały się tzw. exploity, przez co użytkownicy nie mający doświadczenia technicznego (tzw. script kiddies) mogli się włamywać do systemów przez sieć.
- W Linuksie istnieje możliwość ładowania programów i bibliotek pod losowo wybrany adres, co niezwykle utrudnia konstruowanie exploitów.
- Z ostatecznym (???) ratunkiem pośpieszyły Intel i AMD: najnowsze wersje ich procesorów mają możliwość określenia zabronienia zakazu wykonania kodu (oprócz istniejącego już zakazu zapisu) dla każdej strony procesu.
  - Nic prostszego jak ustawić bit zakazu dla wszystkich stron stosu.
  - Zamieniamy możliwość włamania się do systemu na atak DoS (usługa spowoduje błąd stronicowania i przestaje działać).
- Nowe języki programowanie (Java, C# - platforma .NET) nie używają wskaźników - i problemu nie ma (???)

Słynne luki w mechanizmach bezpieczeństwa

- Program lpr drukujący plik na drukarce w Uniksie wykonuje się z przywilejami superużytkownika (ustawiony bit suid). Ma on opcję pozwalającą na skasowanie drukowanego pliku. We wczesnych wersjach systemu możliwe było wydrukowanie i skasowanie pliku z hasłami (/etc/passwd).
- Polecenie mkdir (ustawiony bit suid) najpierw tworzyło i-węzeł katalogu (wywołanie systemowe mknod) a następnie zmieniało jego właściciela z root (effective UID) na użytkownika wywołującego program mkdir (real UID) przy pomocy funkcji chown. W czasie pomiędzy wywołaniami mknod a chown proces użytkownika mógł (a) usunąć i-węzeł (b) utworzyć dowiązanie do pliku /etc/passwd. W rezultacie użytkownik stawał się właścicielem pliku z hasłami.
- System TENEX (maszyna DEC-10) pozwalał na zabezpieczenie pliku hasłem. Aby otworzyć taki plik proces powinien dostarczyć wskaźnik na tekst hasła. TENEX wykorzystywał stronicowanie, a proces mógł się wiedzieć, gdy powstał wyjątek stronicowania.
  - Hasło sprawdzane znak po znaku, pierwszy błędny znak daje błąd ILLEGAL_PASSWORD

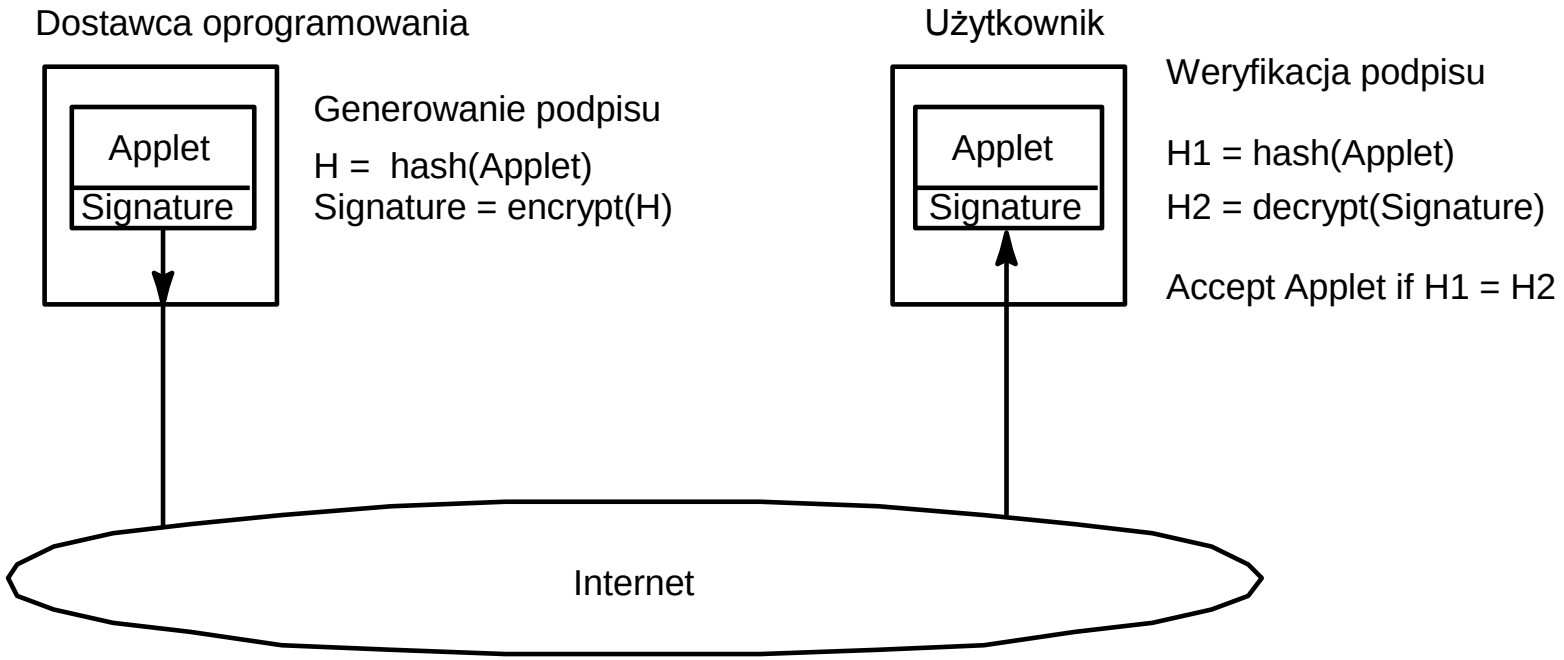
Włamanie do TENEX'a



- Zapewnij, że jedna ze stron jest w pamięci a drugiej nie ma (np. generując odpowiedni ciąg odwołań). Umieść hasło tak, aby pierwsza litera była tuż przed granicą stron.
- Spróbuj otworzyć plik (a), jeżeli otrzymałeś komunikat ILLEGAL_PASSWORD i wykryłeś błąd stronicowania, to odgadłeś pierwszą literę i przejdź do zgadywania drugiej (c). Jeżeli błędu stronicowania nie było to zmień literę na następną (b)
- W ten sposób K-literowe hasło można odgadnąć przy pomocy 128*K prób, a nie 128^K prób.

Kod mobilny (applety, Active X)

- Czasami z pewnych względów musimy uruchomić kod przesłany z sieci, któremu nie ufamy do końca (np. skaner antywirusowy on-line).
- Stosowane są następujące rozwiązania:
  - uruchom kod w interpreterze (np. wewnątrz przeglądarki). Interpreter pozwala na sprawdzenie czy kod używa właściwych adresów oraz na przechwycenie wywołań systemowych. Przykładem jest wirtualna maszyna Javy wewnątrz przeglądarki.
  - Kod mobilny jest podpisany przez organizację której wszyscy ufamy :))
  - W przykładzie , dla zwiększenia wydajności, zastosowano połączenie jednokierunkowej funkcji skrótu z kryptografią asymetryczną



Bezpieczeństwo Javy (1)

- Java jest językiem z bezpiecznym systemem typów.
  - Kompilator nie pozwoli na użycie zmiennej niezgodnie z jej przeznaczeniem.
  - Brak typu wskaźnikowego.
- Kompilator generuje kod dla maszyny wirtualnej (JVM), interpreter przed uruchomieniem sprawdza czy kod przestrzega następujące reguły (intruz mógłby napisać złośliwy kod używając assemblera JVM):
  - usiłowanie zasymulowania wskaźników
  - usiłowanie odwołania się do prywatnych pól klasy
  - usiłowanie użycia zmiennej jednego typu jak zmiennej długiego
  - generowanie przepełnień stosu.
  - nielegalna konwersja zmiennych.
- Problem jest z dostępem do obiektów np. plików. Mechanizm ochrony jest bardzo drobnoziarnisty.

Bezpieczeństwo Javy (2)

URL	Signer	Object	Action
www.taxprep.com	TaxPrep	/usr/susan/1040.xls	Read
*		/usr/tmp/*	Read, Write
www.microsoft.com	Microsoft	/usr/susan/Office/	Read, Write, Delete

- Użytkownik może zdefiniować politykę bezpieczeństwa składającą się z reguł postaci:
  - URL - skąd sprowadzono applet
  - Signer - kto (klucz publiczny) go podpisał.
  - Object - podzbiór drzewa katalogów, adres IP, adres DNS.
  - Action (w przypadku komputerów sieci prawo akceptacji i nawiązania połączenia).
- Ponadto dużo, dużo więcej np. możliwość ścisłej kontroli skąd applet może ładować kod dodatkowych klas.

Wykorzystanie inżynierii socjalnej

- Użytkownik dostaje (sfingowany) e-mail ze swego banku informujący o włamaniu i proszący o natychmiastowe zalogowanie się i zmianę hasła. Dla wygody e-mail zawiera link do strony banku.
  - Tylko to nie jest autentyczna strona banku.
- Administrator dostaje (sfingowanego) e-maila z serwisu technicznego informujący dziurze w systemie i konieczności ściągnięcia poprawki (spod wskazanego URLa).
- Zwycięzca aukcji dostaje (sfingowany) e-mail od sprzedawcy z numerem konta na który należy wysłać pieniądze.
- Walka z takimi metodami jest niezmiernie trudna, ponieważ wymaga ***edukacji*** ludzi.
  - np. personel banku wie, że każdy e-mail od pomocy technicznej musi zawierać specyficzne hasło.
  - Dygresja: Jaką pensję powinien wyznaczyć dyrektor banku dla informatyka ? Odp. taką samą jak dla zastępcy dyrektora.

Zasady projektowania bezpiecznych systemów

- Budowa systemu powinna być powszechnie znana.
- Domyślnie, dostęp powinien być zabroniony.
- Sprawdzaj uprawnienia na bieżąco.
- Dawaj procesom jak najmniejsze uprawnienia.
- Mechanizmy ochrony powinny być proste, jednolite i zaimplementowane w najniższych warstwach systemu.
- Wybrane metody ochrony nie mogą zniechęcać użytkowników.

...i utrzymuj prostotę budowy