

Ataki na systemy informatyczne - analiza zagrożeń

Przemysław Frasunek, Paweł Pisarczyk

venglin@freebsd.lublin.pl

pawel.pisarczyk@ii.pw.edu.pl

Plan prezentacji

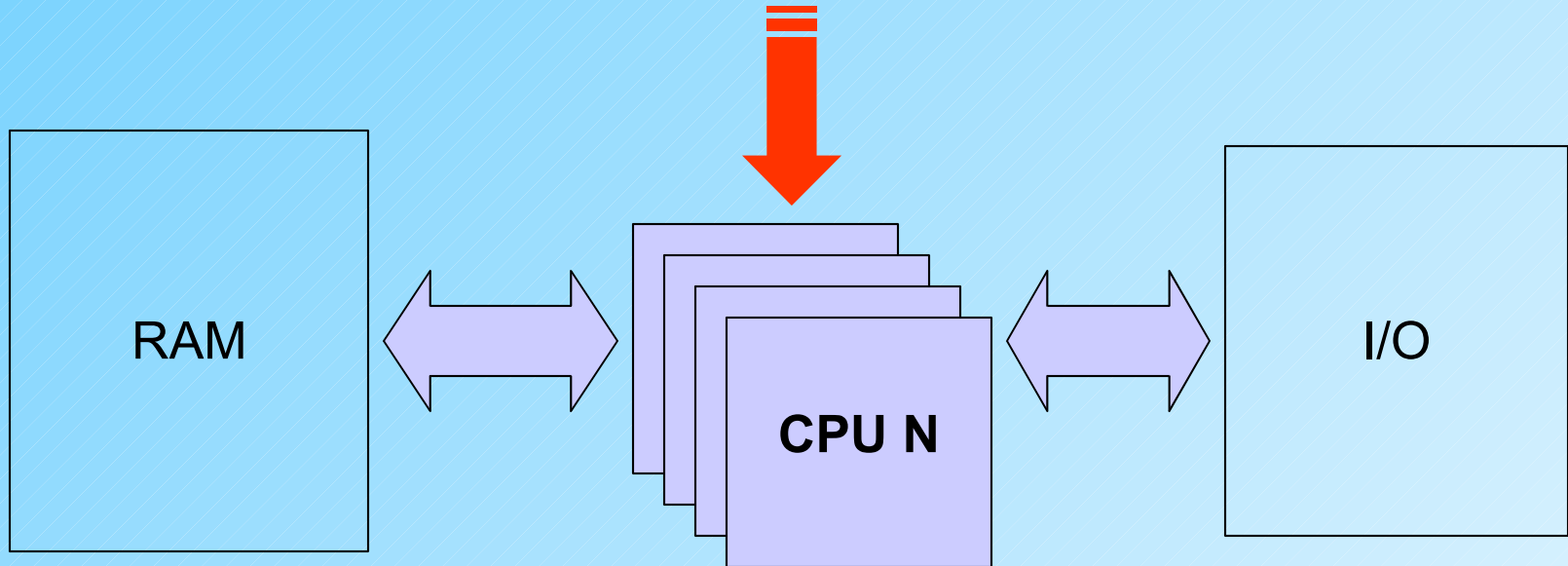
- Wstęp
- Błędy implementacyjne
- Ataki
- Metody ochrony
- Wiarygodność oprogramowania
- Podsumowanie

Hacker - definicje

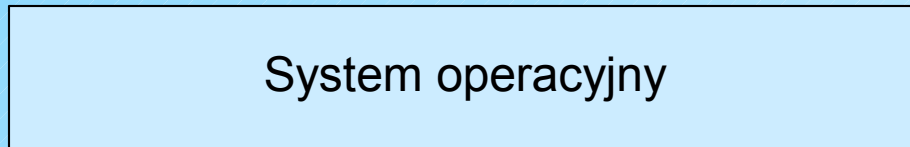
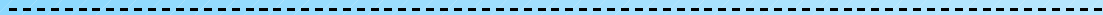
- Osoba podejmująca intelektualne wyzwanie kreatywnego i błyskotliwego ominięcia ograniczeń stosowanej technologii.
- Doświadczony programista, posiadający intuicję i głęboką, szczegółową wiedzę na temat architektury systemów komputerowych.
- **W mediach termin ten stosowany jest błędnie jako synonim przestępcy komputerowego!**

Model systemu komputerowego (1)

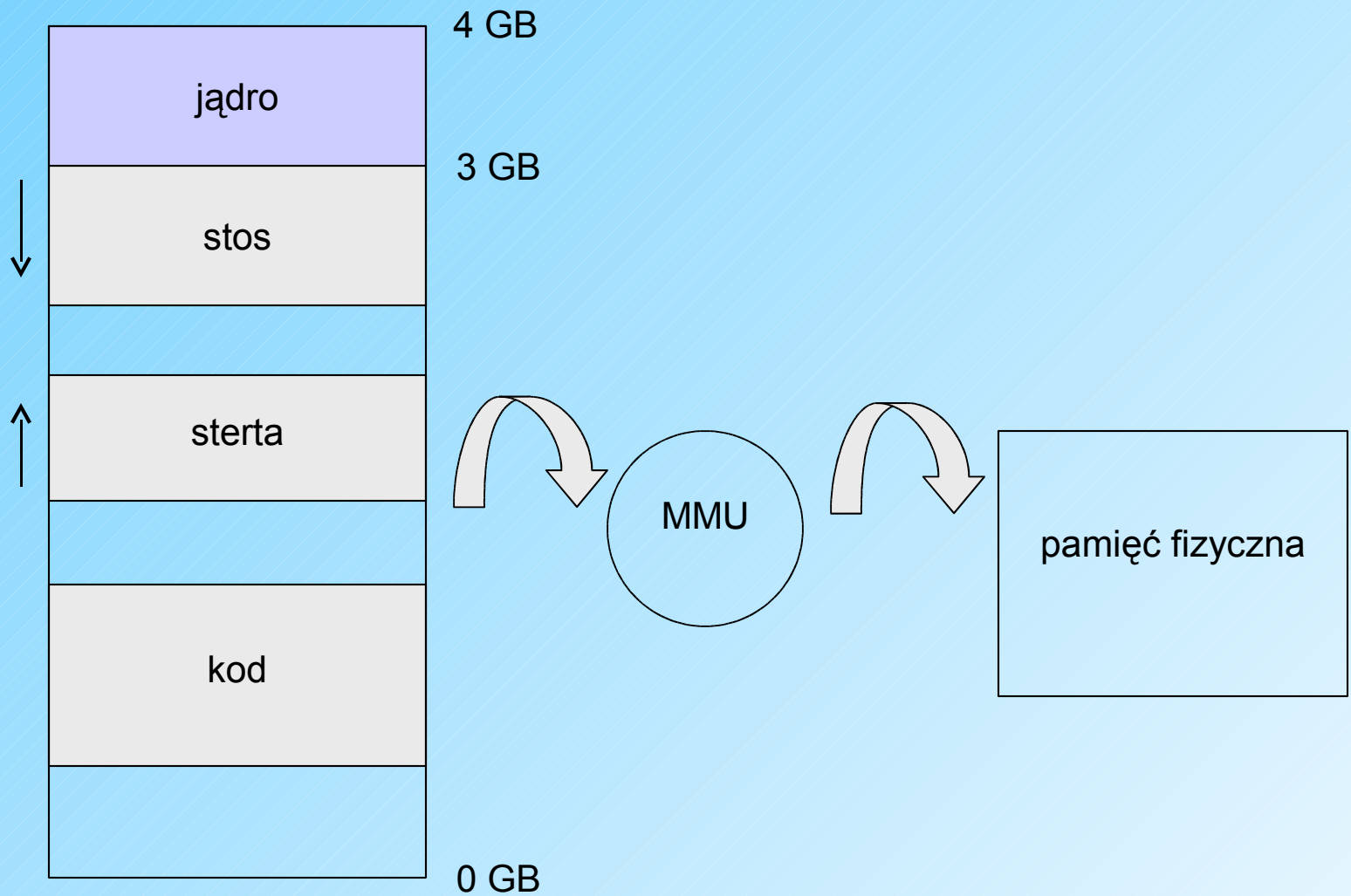
przerwania



Model systemu komputerowego (2)



Przestrzeń adresowa procesu



Stronicowanie

- Popularny mechanizm odwzorowania pamięci wirtualnej w fizyczną.
- Odwzorowanie odbywa się za pomocą tablicy stron przez jednostkę zarządzania pamięcią tzw. MMU.
- Element tablicy stron wiąże adres fizyczny z wirtualnym i zawiera atrybuty strony (strona do odczytu, do zapisu, wykonywalna, systemowa).
- W większości architektur nie istnieje oddzielny atrybut wykonywalności.
- Na poziomie stronicowania realizowana jest ochrona jądra systemu operacyjnego i separacja procesów.

Model uprawnień w systemach operacyjnych

- Podział systemu na jądro (warstwę zaufaną) i aplikacje użytkowe.
- Jądro realizuje założony model bezpieczeństwa (integralność jądra chroniona jest sprzętowo).
- Dyskretny model bezpieczeństwa - semantyka „wszystko, albo nic”.
- Uprawnienia są weryfikowane tylko w oparciu o typ użytkownika i związane z nim przywileje.

Specyfika ataków

- Ataki wykorzystują błędy implementacyjne w udostępnionych usługach i aplikacjach.
- Błędy implementacyjne są nieuchronne.
- Aplikacja z reguły nie może zostać przetestowana w sposób wyczerpujący.
- Błędy w systemie operacyjnym rzutują na bezpieczeństwo pozostałych komponentów.

Błędy implementacyjne

- Błędy typu „buffer overflow”
- Błędy typu „formatting bugs”
- Odporność na ataki DoS (Denial of Service)
- Błędy w aplikacjach WWW
- Hazard czasowy, wyścigi

Błędy typu “buffer overflow” (1)

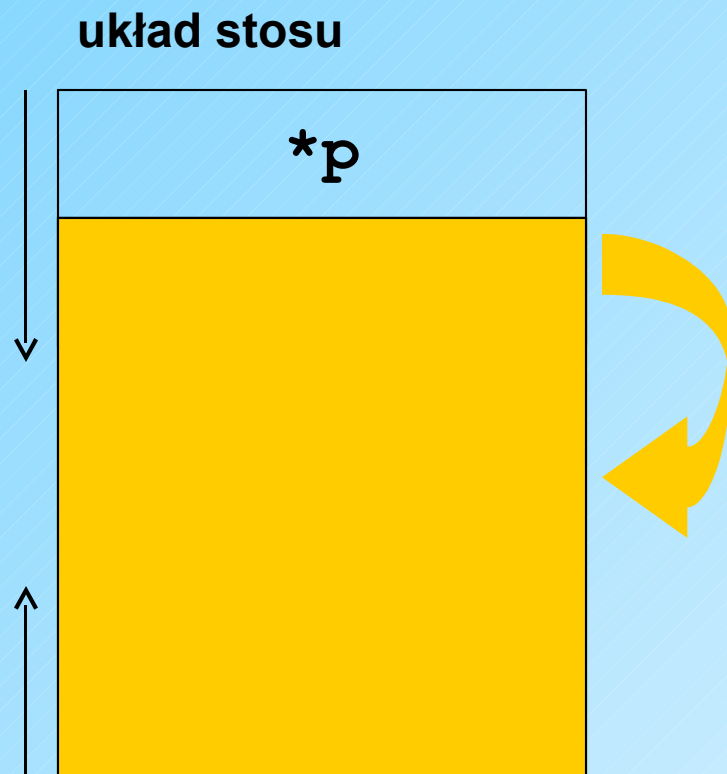
- Buffer overflow – błędy wykorzystywane od 1988 roku polegające na kontrolowanym nadpisaniu przestrzeni adresowej procesu (np. stosu lub sterty) danymi, mogącymi zmienić przebieg wykonania procesu, np. poprzez modyfikację adresu powrotu funkcji lub wskaźnika do innej funkcji w tablicy GOT. Istnieje bardzo wiele odmian tego typu błędów, np.:
 - klasyczne: niewłaściwe wykorzystanie funkcji operujących na łańcuchach tekstowych – strcpy(), strcat, sprintf(), itd.
 - one-byte overflow – polega na nadpisaniu najmłodszego bajtu zachowanego rejestru %ebp
 - sprytne wykorzystanie braku terminacji łańcucha bajtem zerowym
 - brak lub niewłaściwy warunek brzegowy w pętli
 - i wiele innych

Błędy typu “buffer overflow” (2)

```
void funkcja(char *p)
{
    char buf[1024];

    strcpy(buf, p);
    return;
}
```

Błędy typu “buffer overflow” (3)



Błędy typu “integer overflow”

- Integer overflow – błędy powszechnie znane zaledwie od kilku lat, polegają wykorzystywaniu oczywistej cechy zmiennych całkowitych ze znakiem: na platformie 32-bitowej przekroczenie wartości $+2^{31}-1$ powoduje zmianę wartości na -2^{31} ; wykorzystanie ujemnej zmiennej jako argumentu do funkcji `malloc()` NIE powoduje błędu, a jedynie zaalokowanie kilkubajtowego bufora, który może być z łatwością nadpisany

Błędy typu “formatting bugs” (1)

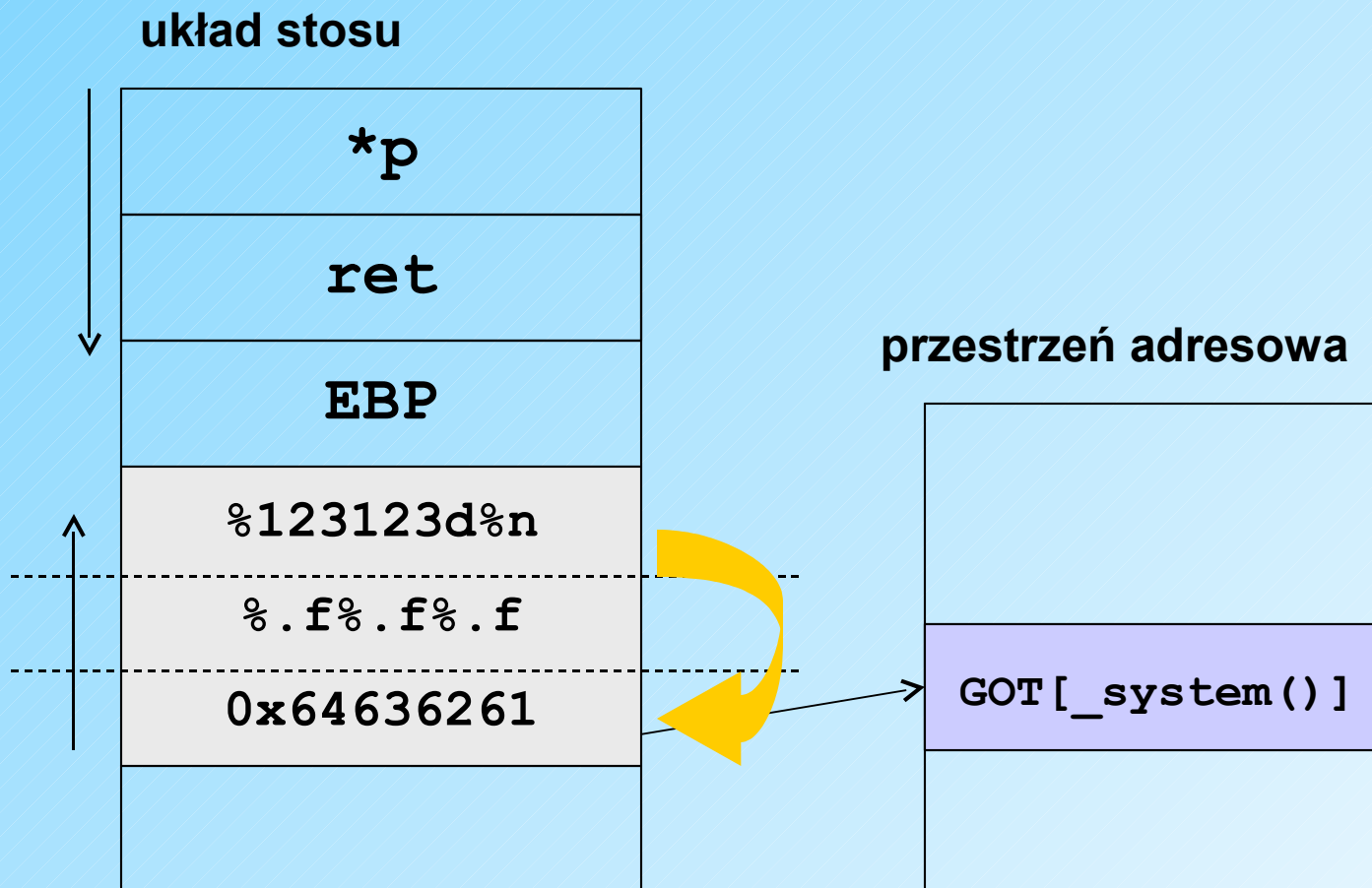
- Format string bugs – metodologia wykorzystywania tych błędów została opracowana przez autora wykładu w 2000 roku, wynikają one z pominięcia przez programistę łańcucha formatującego przy wykorzystaniu funkcji z rodziny printf(). Taki błąd nie jest sygnalizowany przez kompilator C, a umożliwia napastnikowi nadpisywanie dowolnych komórek pamięci w przestrzeni adresowej procesu

Błędy typu “formatting bugs” (2)

```
void funkcja(char *p)
{
    char buf[1024];

    snprintf(buf, p);
    return;
}
```

Błędy typu “formatting bugs” (3)



Odporność na ataki DoS

- Systemy ogólnego przeznaczenia są sterowane zdarzeniami.
- Podatność na ataki DoS wynika najczęściej z błędów projektowych lub błędów konfiguracyjnych.
- Twórcy oprogramowania aplikacyjnego i administratorzy nie są świadomi ograniczeń technologicznych.
- Tworzone systemy informatyczne nie są poddawane testom obciążeniowym i wydajnościowym.
- **Panuje przeświadczenie, że przed atakiem DoS nie można się obronić.**

Błędy aplikacji WWW

- Błędy aplikacji WWW – XSS
- Błędy aplikacji WWW – SQL injection

```
mysql_query("SELECT * FROM tabela WHERE id='$zmienna'");
```

```
http://www.host.com/skrypt.php?zmienna='OR 1;'
```

Komentarz dotyczący bezpieczeństwa w roku 2003 (1)

- Aplikacje OpenSource

- rośnie świadomość programistów
- praktycznie nie występują już “proste” błędy typu buffer overflow
- błędy typu “formatting bugs” również zostały praktycznie całkowicie wyeliminowane
- rośnie znaczenie błędów typu integer overflow i bardziej ukrytych buffer overflow'ów, których nie można wykryć podczas automatycznego audytu kodu
- poprawione wersje większości aplikacji pojawiają się najpóźniej w ciągu kilkudziesięciu godzin od opublikowania informacji o błędzie

- Aplikacje zamknięte

- brak konieczności udostępniania kodu źródłowego powoduje, że programiści niejednokrotnie nie dbają o styl i poprawność kodu

Komentarz dotyczący bezpieczeństwa w roku 2003 (1)

- w dalszym ciągu do najbardziej popularnych błędów należą klasyczne buffer overflow, czasami również format string bugs
- ciekawy przykład: niebezpieczne fragmenty kodu zostały przytoczone w MSDN jako całkowicie poprawne!
- fatalny czas reakcji na informacje o błędach: niejednokrotnie kilka miesięcy (Microsoft), a nawet rok (Oracle)!
- Statystyki liczby błędów w popularnych systemach operacyjnych:
 - FreeBSD (domyślnie zainstalowany system, wszystkie usługi):
 - 6 remote root (3 w domyślnie uruchomionych usługach)
 - 4 słabości kryptograficzne (1 dotyczy kernela)
 - 4 remote DoS (1 dotyczy kernela)

Komentarz dotyczący bezpieczeństwa w roku 2003 (2)

- 4 local DoS w kernelu, które dodatkowo w szczególnych warunkach mogą spowodować ujawnienie fragmentu pamięci jądra

– OpenBSD:

- 6 remote root (żaden w domyślnie uruchomionych usługach)
- 3 local root (wszystkie w kernelu)
- 5 słabości kryptograficznych/algorytmicznych
- 9 remote DoS (5 w kernelu)
- 5 local DoS (4 w kernelu)

– Linux (tylko kernel):

- włamanie do repozytorium BitKeepera i podłożenie konia trojańskiego
- 1 local root
- 3 remote DoS
- 4 local DoS

Komentarz dotyczący bezpieczeństwa w roku 2003 (3)

– Windows

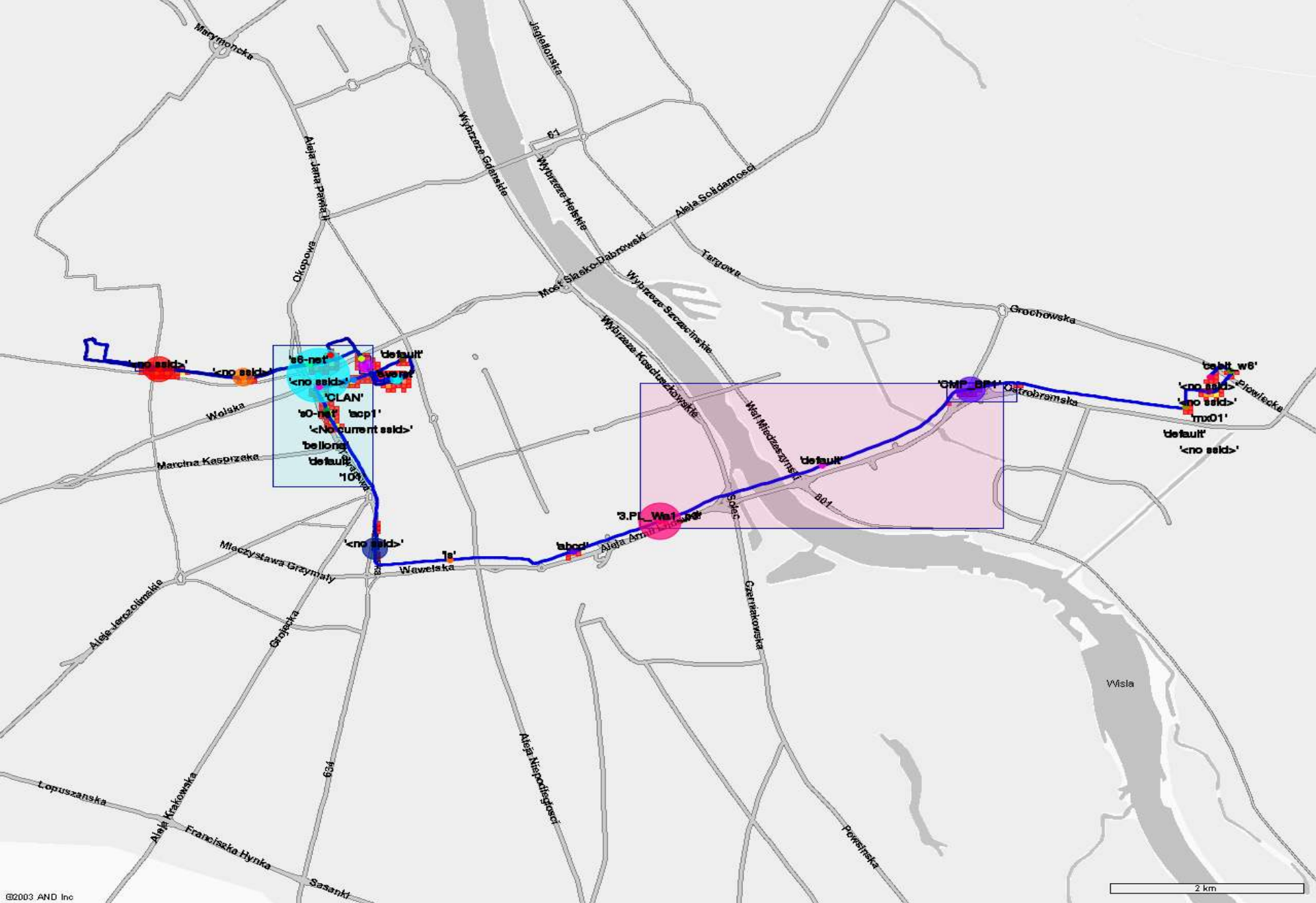
- 27 błędów umożliwiających zdalne wykonanie kodu
- 4 błędy umożliwiające lokalne wykonanie kodu z podwyższonymi uprawnieniami
- Wyciek części kodu źródłowego nie spowodował znacznego przyrostu w liczbie wykrywanych błędów

Główne zagrożenia w roku 2003 (1)

- Duża liczba błędów w systemach z rodziny Windows, niektóre z nich umożliwiają zdalne wykonanie dowolnego kodu z uprawnieniami SYSTEM bez żadnej ingerencji użytkownika
- Kilkadziesiąt błędów w Microsoft Internet Explorer i Microsoft Outlook
- Kilka poważnych błędów w aplikacji Sendmail, która jest najpopularniejszym MTA
- Poważny błąd DoS w systemie operacyjnym IOS zainstalowanym na routerach Cisco
- Kilka błędów w Sun RPC, które dotyczą wielu systemów operacyjnych z rodziny Unix

Główne zagrożenia w roku 2003 (2)

- Pojawienie się robaków internetowych, wykorzystujących błędy w systemie Windows
- Robaki wykorzystywane do ataków typu DDoS (np. Blaster, SDBot) i rozsyłania spamu
- Włamanie do serwerów projektów OpenSource – Debian, Gentoo, GNU, Kernel BitKeeper
- Rosnąca popularność aplikacji WWW i łatwość ich tworzenia powoduje upowszechnienie błędów typu SQL injection, XSS, czy wykorzystujących metaznaki
- Rozpowszechnienie sieci bezprzewodowych – przejazd samochodem po głównych ulicach Lublina ujawnia ponad 60 sieci 802.11b/g, z których zaledwie kilka używa WEPa i żadna IPSEC!
- Rozpowszechnienie urządzeń Bluetooth – liczne błędy w implementacjach stosu BT w urządzeniach przenośnych



©2003 AND Inc
 Map Coordinates : 52.228796,21.032802 @ scale 30000
 Visible networks: 25
 Map Created : Tue May 18 22:57:20 2004



Włamanie do systemu

- **Zdalna analiza topologii i udostępnionych usług:**
 - metody pasywne,
 - metody aktywne.
- **Identyfikacja podatności na atak:**
 - określenie usług posiadających znane błędy implementacyjne,
 - poszukiwanie nowych błędów i metod włamania.
- **Opracowanie metody ataku**
- **Włamanie**
- **Zamaskowanie obecności w systemie:**
 - użycie pakietu rootkit,
 - oczyszczenie dzienników.

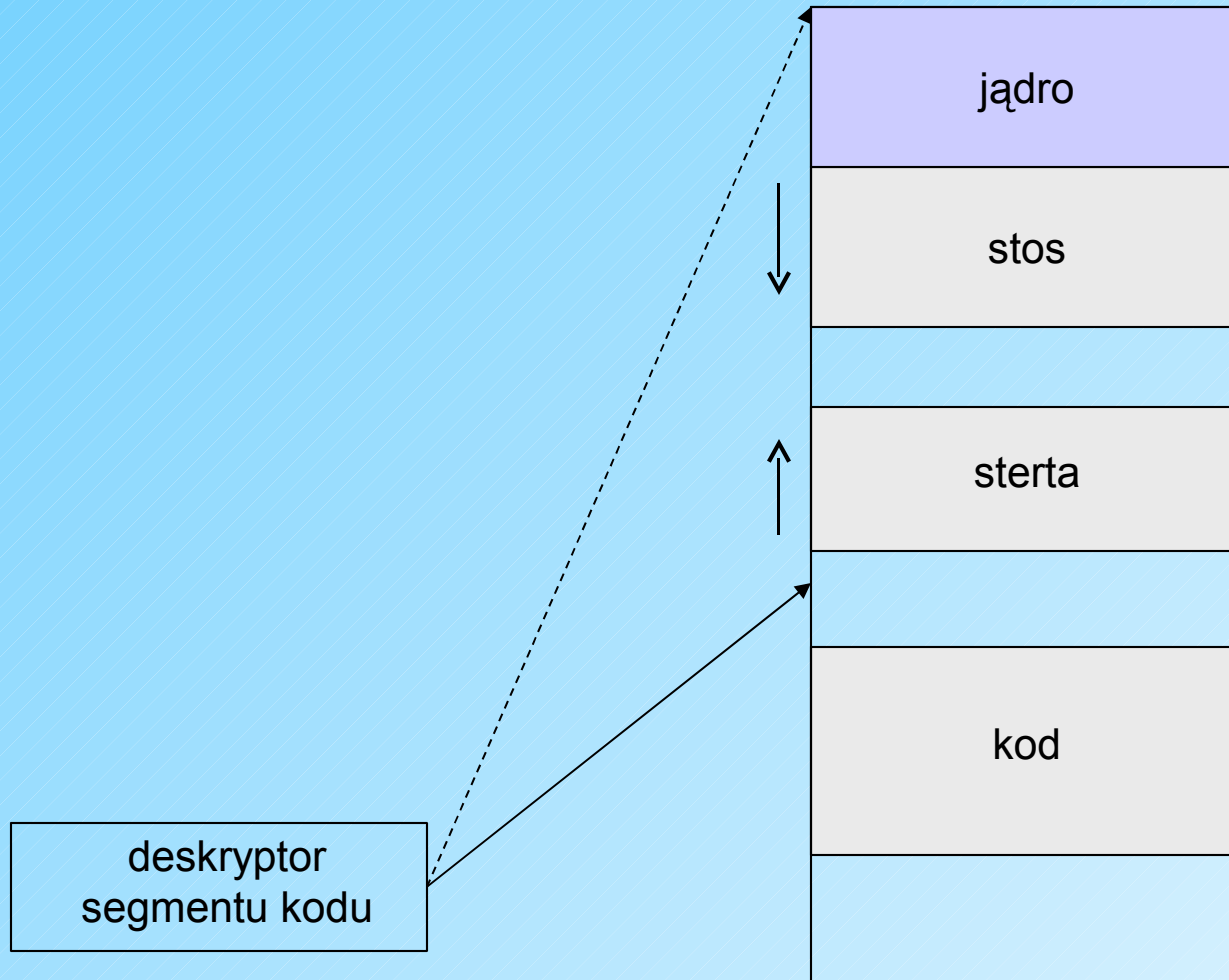
Techniki maskowania

- Skanowanie typu Stealth
- Skanowanie pasywne
- Technika IDS wakeup
- Stosowanie rootkitów, działających na poziomie jądra systemu.
- Wykorzystywanie niestandardowej komunikacji np. ICMP (echo request i echo reply) lub przesyłanie niektórych danych jako numerów sekwencyjnych.

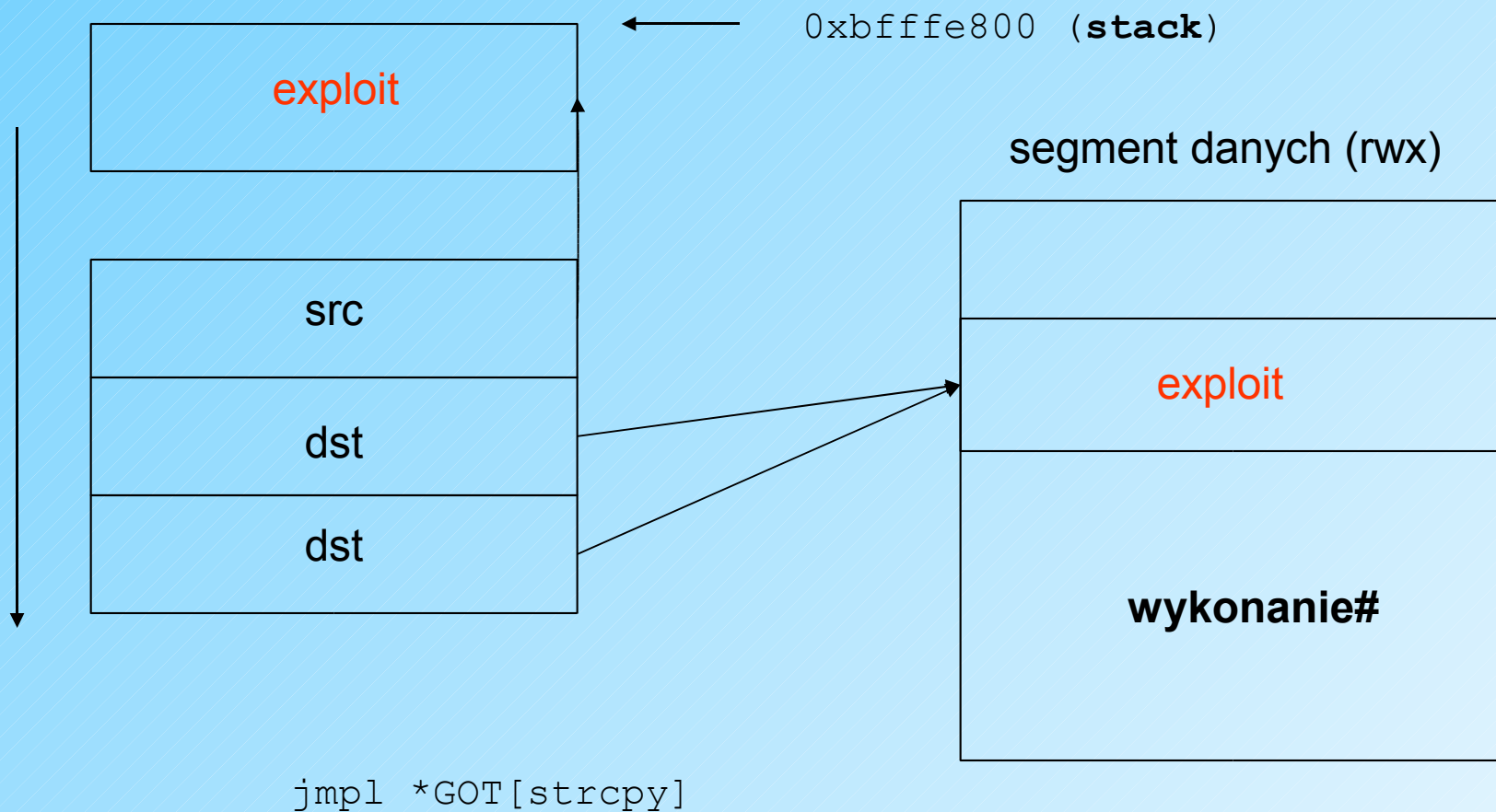
Ochrona przed atakami na poziomie systemu operacyjnego

- **Analiza statyczna (audyt) kodu źródłowego**
- **Ochrona stosu:**
 - ochrona na poziomie kompilacji (StackGuard),
 - ochrona wykorzystująca specyficzne cechy procesora (Solar Patch, Pagexec).
- **Systemy klasy Secure i Trusted:**
 - rozszerzenie standardowego modelu uprawnień,
 - klatki uprawnień.

Ochrona stosu Solar Patch IA32



Metoda omińnięcia Solar Patcha (zaproponowana przez R. Wojtczuka)

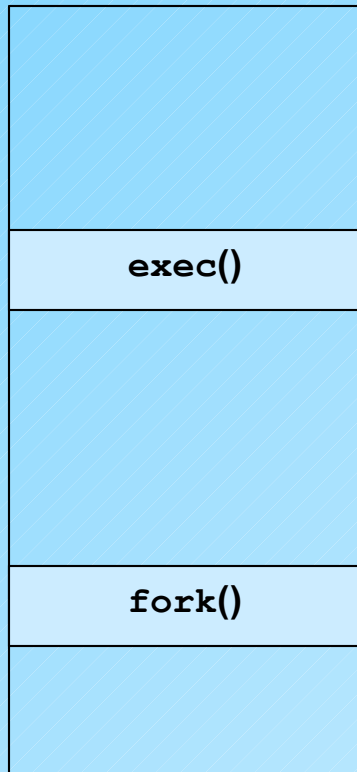


Systemy Trusted

- Dostarczają funkcjonalność systemu ogólnego przeznaczenia.
- Można stosować w nich komponenty programowe spotykane w klasycznych systemach.
- Model bezpieczeństwa definiowany jest na poziomie systemu i może zostać efektywnie narzucony aplikacjom.
- Wykonujące się procesy mogą zostać od siebie ściśle odseparowane.

Systemy Trusted - autoryzacja wywołań systemowych

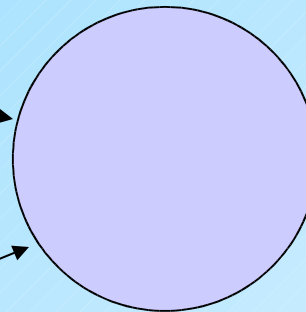
Aplikacja



Serwer uprawnień

REJECT

ALLOW



Wiarygodność oprogramowania

- Oprogramowanie nie jest wyczerpująco testowane.
- Nie istnieje polityka jakości, dotycząca oprogramowania.
- Oprogramowanie jest tworzone przy założeniu poprawnego działania sprzętu.
- Istnieje klasa aplikacji, dla których warunek ten nie jest wystarczający (np. aplikacje lotnicze).
- Niektóre systemy promowane są jako wiarygodne, mimo braku analizy wiarygodności.

Podsumowanie

- Błędy implementacyjne stanowią główne źródło zagrożeń systemów informatycznych.
- Błędy implementacyjne są nieuchronne (złożoność oprogramowania, brak testów, czynnik ludzki).
- Ochrona przed atakami jest procesem ciągłym, stanowiącym połączenie wiedzy i technologii.
- Audyt, podnoszenie kwalifikacji, monitorowanie i analiza zagrożeń pozwalają na efektywne zmniejszenie ryzyka ataku.