

# Dafne - Distributed Angels for Network Environments

Paolo Perego  
Sikurezza.org  
sponge@sikurezza.org  
Ottobre 2004, Milano, Italy

## Abstract

*Il limite del progetto AngeL[1], che mira a rendere un host incapace di iniziare un attacco informatico contro terzi, è quello di non prendere in considerazione i nodi adiacenti all'elaboratore in questione in un contesto di rete. In realtà dove sempre di più occorre rendere la rete un luogo "sicuro" nel quale lavorare ed interagire coi vari servizi scambiandosi informazioni, diventa importante aumentare il grado di protezione dei nostri elaboratori ed operarci per prevenire un attacco informatico. Unendo questa necessità alla flessibilità introdotta dalla programmazione tramite agenti mobili, si studierà un sistema cooperativo in grado di reagire attivamente ad un attacco informatico attraverso un costante monitoraggio dello stato dei singoli nodi di una rete comunque complessa.*

## 1 Introduzione

L'approccio classico circa la sicurezza informatica è quello di proteggere il nostro piccolo (ma comunque prezioso) orticello da chiunque voglia accedervi non essendo in possesso delle credenziali necessarie per farlo. Non solo, un buon amministratore di sistema prenderà tutte le precauzioni affinché un qualsiasi maleintenzionato impedisca

ad utenti legittimi di accedere alla verdura del nostro orticello. Questo è quanto. Il problema della sicurezza informatica viene quindi affrontato riducendo al minimo<sup>1</sup> la possibilità di un'intrusione o comunque di un intervento dall'esterno atto a violare la rete stessa o uno qualsiasi dei nodi ad essa connessi.

Nel luglio del 2001, nato come lavoro di evoluzione di HOT-I, viene rilasciato il progetto AngeL. Del progetto pilota, AngeL, mantiene la filosofia di base: l'approccio alla sicurezza informatica non doveva più essere visto come un limitare le possibilità che un attacco possa venir portato con successo, bensì limitare le capacità di un elaboratore, agendo sul kernel di sistema operativo, di iniziare un attacco informatico.

Il progetto AngeL implementa questo nuovo approccio attraverso un modulo kernel per il sistema operativo linux[2] vincolo questo troppo restrittivo se vogliamo portare questo nuovo approccio alla sicurezza in un ambiente di rete eterogeneo<sup>2</sup>. Dovremo quindi cercare una soluzione

---

<sup>1</sup>In seguito non menzionerò ulteriormente questo concetto. A meno di non tenere un elaboratore scollegato da qualsiasi rete e privo di qualsiasi dispositivo di input, la sicurezza informatica al 100% non è raggiungibile. Si può solo aumentare il grado di robustezza dei nostri sistemi.

<sup>2</sup>Si obietterà che un attaccante con molta probabilità non installerà tali contromisure sulla propria macchina. Questo è vero nel momento in cui non riusciamo a fonde-

software che sia eterogenea quindi totalmente indipendente dal sistema operativo e dall'architettura sottostante. Per semplicità, in questa prima fase di studio del progetto, supporremo che i sistemi operativi dei nodi della rete siano di una qualsiasi famiglia di unix.

Il modello che meglio descrive un'entità dinamica che si astragga dall'architettura e dalla tipologia dell'infrastruttura sottostante, è quello degli agenti mobili. Quindi utilizzeremo gli agenti mobili per creare una rete cooperativa in grado di monitorare costantemente l'attività della rete e degli host collegati.

## 2 Dafne: infrastruttura operativa

L'idea di mobilità introdotta dal concetto di agente (vedremo più avanti come formalmente si stia introducendo un meccanismo di agenti cooperativi) si scontra con la sua implementazione in un software che ne modelli tali caratteristiche. In particolare il primo problema da affrontare è quello di implementare un meccanismo che consenta ad un processo, inteso come raffigurazione software del modello di agente, di replicarsi e spostarsi su nodi generici di una rete (analizzeremo in seguito le caratteristiche di pseudo intelligenza artificiale che l'agente deve possedere).

L'idea che meglio sposa la necessità operativa di avere dei processi in grado di spostarsi autonomamente da un host all'altro, è quella di avere una serie di demoni residenti su vari host della rete. La presenza di un demone per ogni nodo del-

---

re tali contromisure col sistema operativo stesso. Tuttavia, qualora non riuscisse questa fusione, eviteremo che l'attaccante usi altre macchine come teste di ponte per altri attacchi

la rete non è mandatoria, tuttavia assicurerebbe la certezza della completa esplorabilità della rete da parte dell'agente. Il problema del routing dell'agente all'interno della rete sarà affrontato assieme all'analisi dell'agente stesso (sezione 3), tuttavia appare chiaro che sui nodi nei quali non è presente il demone nessun agente potrà accedervi.

Stiamo disegnando quindi una rete di processi server nei quali il transito di informazioni attraverso gli usuali metodi di comunicazione tra processi (socket, pipe, rpc, ...) modella l'agente in transito. Ogni demone di sistema sarà responsabile degli agenti in esecuzione sull'host nel quale è in esecuzione, sarà inoltre responsabile della migrazione dell'agente dal proprio host ad un'altra macchina della rete. L'agente avrà il compito di implementare dei security check seguendo un elenco fornito all'agente durante la creazione dello stesso. Ipotizziamo quindi che sia possibile creare agenti differenti fornendo a ciascuno una lista di controlli differenti da svolgere sulla base di un elenco di controlli comune. L'agente in realtà chiederà al demone di eseguire tali controlli per lui e di comunicare i risultati, il demone infatti viene visto come unico tramite tra l'agente e il sistema operativo. Sulla base dei risultati acquisiti da questo pool di security check, l'agente basandosi su euristiche e sui dati provenienti sia da altri agenti che dall'analisi effettuata eventualmente su altri elaboratori, è in grado di stabilire se ci si trovi in una situazione di normalità o in una situazione di pericolo. Nel caso l'agente rivelasse che è in corso un tentativo di attacco sia diretto verso l'elaboratore sotto analisi che generato dallo stesso elaboratore, comunicherà al demone le contromisure da prendere. Un'ultima ipotesi è quindi quella di stabilire un set di regole per descrivere in maniera univoca come reagire ad un determinato attacco e soprattutto

come implementare i test di sicurezza da condurre. Questo set di regole sfocierà in un linguaggio descrittivo che sia il demone che l'agente saranno in grado di comprendere ed utilizzare<sup>3</sup>.

L'ambiente operativo nel quale ci poniamo è quello, per semplicità, di una LAN comunque complessa. Potranno essere studiate estensioni a questo modello che trattino reti geografiche la cui topologia non è nota a priori oppure Internet stessa come caso estremo.

Scenderemo nel dettaglio nelle sezioni 3 e 4 per descrivere rispettivamente le entità mobili del sistema e le entità statiche.

### 3 L'entità mobile: l'agente

L'agente è un processo. All'interno di una rete, su ogni singolo host, saranno in esecuzione un numero arbitrario di agenti per monitorare le attività dei vari elaboratori evidenziando istantaneamente comportamenti anomali del sistema. La presenza di almeno un agente su ogni nodo non è mandatoria, tuttavia i nodi sui quali non è presente in quel momento un agente sono esclusi dall'analisi in tempo reale dei vari attacchi. La presenza del demone `dafned` garantisce che messaggi di aggiornamento su situazioni pericolose possano giungere anche a questi elaboratori che saranno sempre aggiornati quindi sullo stato del sistema.

Durante l'avvio l'agente effettua una sorta di esplorazione della propria rete per cercare di capire quali elaboratori siano in grado di accettare una sua eventuale richiesta di connessione. L'unica informazione in tal senso che viene fornita

---

<sup>3</sup>Sarebbe addirittura auspicabile che potesse essere utilizzato il linguaggio di regole di `snort` per eseguire i security check e specificare come il demone deve effettuare un *fine-tuning* del sistema operativo per contrastare al tentativo di attacco.

all'agente nel momento della creazione, è l'indirizzo ip del primo nodo da visitare. Partendo da questo indirizzo verranno sondati via via i vari host che l'agente ha scoperto durante la fase di *discovery* in accordo anche con un *tll*<sup>4</sup> assegnato all'agente in maniera esplicita o attraverso un valore di default.

L'agente per potersi trasferire su un elaboratore ed iniziare così le proprie attività, ha bisogno di chiedere l'autorizzazione al demone in ascolto su quell'elaboratore, attraverso la funzione `dafne_agent_handshake()`. Il processo server potrà, a questo punto, accordare o meno la richiesta dell'agente. Nel caso la richiesta venga rifiutata, l'agente passerà a contattare l'host successivo e così via fino ad esaurire tutti gli indirizzi ip dei server presenti in rete. Nel caso la richiesta venga accettata, `dafned` creerà un nuovo processo e vi trasferirà tutti i dati provenienti dall'elaboratore dal quale l'agente sta arrivando. Quello che il server compie è una vera e propria clonazione dell'agente sulla propria macchina.

Tra i dati contenuti nell'agente troviamo, oltre alle credenziali che indicano dove l'agente è stato generato, è contenuto un contatore che indica la *time to live* dello stesso. Verrà impostato un numero massimo di hop realizzabili dall'agente, superato il quale avremo l'eliminazione della sonda dinamica dal sistema.

Il nocciolo dei dati caratterizzanti l'agente è costituito dai risultati dei test acquisiti in precedenza. Il possesso di uno storico, consente all'agente di affinare la probabilità di non incorrere in falsi positivi potendo contare su dati statistici sempre più significativi da utilizzare come base per i confronti con i dati osservati. Insieme a questa base dati, l'agente porta con sé anche i test stessi che deve eseguire, test che saranno

---

<sup>4</sup>Time to live.

scritti in un opportuno linguaggio. Questo linguaggio sarà lo stesso utilizzato dal demone per descrivere le contromisure da prendere in caso di attacco. Sebbene la scelta sarà rimandata in fase di implementazione, il TCL o il linguaggio usato per scrivere le regole di SNORT[3] sono candidati validi che saranno presi in considerazione.

L'agente, quindi, è un'entità completamente indipendente avendo al suo interno sia i compiti che deve svolgere, sia i dati utili per prendere le opportune decisioni. Sarà possibile dotare il server di test aggiuntivi che si vuole vengano eseguiti dall'agente, test aggiuntivi che potranno essere aggiunti a quelli noti all'agente a seconda delle impostazioni del server. Oltre ad essere indipendente, il nostro processo mobile è in grado di apprendere durante il proprio ciclo di vita. Essendo i dati acquisiti dai test condivisi sia col demone che con gli altri agenti, tale conoscenza non viene persa allo scadere del *time to live*.

L'idea alla base della struttura che si vuole realizzare è, infatti, che la condivisione dei risultati dei test è uno strumento efficace per rispondere in maniera efficace ad un attacco in quanto aumenta la bontà statistica del campione di riferimento col quale vengono confrontati i dati osservati.

Se l'agente dovesse osservare un tentativo di intrusione si attiverrebbero prontamente tutta una serie di azioni per proteggere il sistema:

- l'agente comunica al demone che deve reagire ad una certa tipologia di attacco;
- il demone, in accordo con le direttive descritte dall'amministratore della macchina che presiede, prende i dovuti provvedimenti. Qualora il demone non sapesse come reagire perché si tratta di una casistica non contemplata dall'amministratore di sistema, viene

richiesto all'agente di descrivere (sempre attraverso il metalinguaggio col quale vengono anche descritti i test) le contromisure necessarie. Se né il demone, né l'agente sanno come intervenire, viene richiesto agli altri demoni/agenti della rete. Se nessuno sa come intervenire, il tentativo viene registrato nei log e come reazione si esclude la macchina dalla rete se si tratta di un tentativo remoto oppure si prendono altri provvedimenti drastici nell'ipotesi di intrusione locale (cambio di runlevel, shutdown del sistema, ...);

- l'agente, avuta conferma dal demone che le contromisure sono entrate in funzione, comunica i risultati della propria attività sull'host. Sarà compito del demone effettuare il broadcast di tali risultati.

Una volta effettuati tutti i test che l'agente ha scritto nella propria *todo list* passerà ad esaminare l'host successivo.

In fase realizzativa di un primo prototipo, verrà volutamente ignorato il problema di autenticazione dell'agente da parte del demone **dafned**. In una fase di sviluppo successiva sarà necessario tuttavia affrontare questo problema per evitare che qualsiasi processo possa spacciarsi come agente e richiedere un tuning del sistema per sovvertirne ad esempio l'affidabilità. Un altro fattore che ci spinge ad affrontare il tema dell'autenticazione è quello di riconoscere un agente della rete alla quale fa parte il demone che lo sta ricevendo da un agente, perfettamente legale, ma magari di un'altra realtà<sup>5</sup>. Un meccanismo che

---

<sup>5</sup>Questo fattore trova un riscontro quando verrà a decadere il vincolo di dafne di lavorare in una LAN. Se tuttavia questo vincolo dovesse rimanere anche in una fase successiva a quella di realizzazione di prototipo allora tale fattore decaderebbe... pur rimanendo valida la necessità di autenticare l'agente in qualche maniera.

andrà sicuramente preso in considerazione per validare o meno l'identità dell'agente sarà quello della crittografia a chiave asimetrica con tutte le problematiche di protezione delle chiavi che essa comporta. Nella sezione 5 analizzeremo più nel dettaglio gli sviluppi futuri del sistema.

## 4 L'entità statica: il demone

Se l'agente descritto nella sezione 3 è una parte importante nel sistema in quanto implementa i controlli di sicurezza necessari per perseguire le finalità del progetto, la parte statica di *dafne*, il demone `dafned`, rappresenta le fondamenta di tutto il progetto. Essendo l'entità servente che accoglie l'agente e che ne permette il successivo trasferimento su un altro elaboratore, oltre a cooperare con esso durante la fase di security check, implementare un demone che sia:

- robusto;
- flessibile in termini di servizi offerti all'agente;
- integrato col sistema operativo ed in grado quindi di operare realmente un tuning efficace del nostro elaboratore;

ci permette di avere solide basi sulle quali poggiare le parte mobile di *dafne*.

Il ciclo di vita del processo server inizia con una fase di discovery della propria rete locale del tutto analoga a quella operata dall'agente, in particolare verrà effettuata la stessa chiamata alla routine `dafne_network_discovery()`.

Una volta individuati gli indirizzi ip sui quali è in ascolto un'istanza di `dafned`, viene annunciato con un broadcast l'avvio del nuovo demone. Questa fase è utile in quanto permette ai demoni

di avere una lista dei nodi sui quali poter redirigere gli agenti che, prima di lasciare l'host sul quale sono in esecuzione, confronteranno l'indirizzo ip scoperto nella loro fase di discovery con quelli conosciuti dal demone al quale sta facendo riferimento. In fase implementativa decideremo se l'agente avrà il potere o meno di fare un *override* dell'informazione fornita dal demone sulla sua prossima destinazione. In particolare potremo decidere anche a livello di configurazione software quanto la figura del server sia vincolante per la mobilità dell'agente.

Costruita questa lista conoscitiva di indirizzi ip, il demone attende banalmente che qualche agente richieda la connessione attraverso la funzione `dafne_agent_handshake()`. Scopo di questa fase di richiesta è il permettere o meno al demone di rifiutare la connessione, sia in base ad eventuali `acl`<sup>6</sup> che in base ad un'autenticazione basata sulla crittografia. In questa prima fase tuttavia questa routine restituirà sempre un valore di verità.

Una volta che l'agente è stato autenticato il demone provvederà a creare un nuovo processo eseguendo il codice dell'agente. Il trasferimento dei dati tra il demone (che li ha ricevuti dall'agente in arrivo) al neonato agente avverrà attraverso una comune pipe unix. Una volta trasferiti i dati l'agente sarà totalmente operativo e comincerà le proprie attività. Il processo server resterà quindi in attesa dei risultati dell'analisi compiuta dall'agente e reagirà come descritto nella sezione 3 se dei tentativi di intrusione dovessero essere rilevati.

L'integrazione di `dafned` col sistema operativo sottostante è fondamentale e deve soddisfare a certi requisiti:

- il demone deve essere in grado di impostare

---

<sup>6</sup>Access Control List.

in automatico le regole di firewalling di sistema (in entrata ed in uscita e per ogni scheda di rete) e deve essere in grado di rendere operative le modifiche;

- il demone deve essere in grado di modificare il comportamento a runtime dei processi utente agendo via `sysctl` sui parametri offerti dal sistema operativo;
- il demone deve essere in grado di rilevare moduli `lsm`<sup>7</sup> compliant scritti appositamente per lavorare con `dafne`. Quest'assunzione sarà vera in un futuro diverso da una prima fase di prototipazione; per il sistema operativo linux verrà pensato un framework per scrivere moduli kernel che utilizzando `lsm` (già presente nel kernel stabile dalla versione 2.6.0) che siano in grado di comprendere ordini impartiti da `dafned`. I moduli `AngeL` e `NuSO` saranno presumibilmente i primi moduli ad avere un layer di compatibilità con `dafned`.

I primi punti sono facilmente implementabili in qualsiasi unix studiando in maniera opportuna il linguaggio di descrizione della risposta ad un attacco informatico. L'ultimo punto, che introduce un livello di granularità molto più interessante per quanto riguarda l'approccio di messa in sicurezza della macchina stessa, risulta percorso percorribile solamente per il sistema operativo linux. Essendo `dafne` ed `AngeL` progetti nati con finalità molto simili, quest'ultimo rappresenterà una pista privilegiata nel processo di sviluppo del framework di interfaccia con `lsm`.

---

<sup>7</sup>Linux Security Module, framework introdotto nel kernel 2.6.0 di linux e che semplifica il compito di chi intende sviluppare moduli per il kernel che implementino security check.

Idealmente, come ogni demone di sistema, `dafned` termina la propria esecuzione solamente con l'interruzione dell'attività dell'elaboratore, quindi il ciclo di vita descritto in precedenza si ripete idealmente all'infinito.

## 5 Sviluppi futuri

Terminata una prima stesura prototipale del modello di difesa di una rete sia da aggressioni provenienti dall'esterno che da tentativi di attacco da parte di peer ad essa collegati, si dovranno tenere in considerazione tenute in disparte in questa prima fase di stesura del codice, in particolare andrà rivisto:

- il meccanismo di autenticazione utilizzato da un agente per registrarsi con `dafned`. Un primo banale attacco a `dafne` potrebbe essere quello di iniettare nella rete falsi agenti per dare istruzioni maliziosi ai demoni degli host collegati sovvertendone i canoni di sicurezza;
- il linguaggio per descrizione dei test e delle contromisure. In una prima fase prototipale (dove saranno presenti veramente poche signature di attacchi banali col solo scopo di dimostrare la validità dell'idea) verrà utilizzato un set di regole basate su una grammatica *hardcoded*. Potrebbe essere interessante rendere il sistema abbastanza flessibile da essere in grado di interpretare regole scritte per gli ids più famosi (come Snort[3]). Questo consentirebbe di riutilizzare gran parte del lavoro effettuato da amministratori di sistema sparsi un po' ovunque e che hanno isolato le signature di un gran numero di attacchi e tentativi di exploit. Anche TCL può rappresentare una va-

lida alternativa per descrivere signature di attacchi.

## 6 Conclusioni

Mano a mano che lo sviluppo dell'idea alla base di dafne proseguiva appariva sempre più chiaro che ci si stava muovendo verso un campo di ricerca già noto ed ampiamente coperto dalla letteratura. Quello di utilizzare il modello del sistema immunitario umano come base per studiare metodi di intrusion detection innovativa è argomento che affascina da anni i ricercatori. Dafne, per ora, non ha la presunzione di voler essere la risposta definitiva a questo problema teorico; tuttavia l'idea di entità mobili aventi un certo livello di intelligenza artificiale ed in grado di richiedere l'esecuzione di security check ed istruire il sistema ospite per reagire di conseguenza, porta ad una maggior automazione nel processo di tuning di un host, distribuito su una rete eterogenea. Sebbene l'idea di automatizzare parte del processo di tuning possa far storcere il naso agli amministratori di sistema più meticolosi, si noti come saranno sempre loro ad istruire `dafned` su come reagire di fronte ad un attacco. Avranno inoltre la libertà di lasciare non implementata la routine di risposta nel caso volessero mantenere il pieno controllo.

Per quanto riguarda l'overhead introdotto dai security check, prima di mettere nero su bianco cifre significative si dovrà attendere la stesura del codice del prototipo del demone e dell'agente. Trattandosi comunque di stime sul traffico che l'agente vede transitare sull'interfaccia di rete e banalmente, in un primo momento, di analisi dei dati forniti dal kernel attraverso il file system `/proc`, l'impatto sulle performance del sistema non dovrebbe superare il 5% in termi-

ni di tempo di esecuzione dei processi utente. Questa prima stima è basata sulle analisi effettuate durante l'implementazione di AngeL. Allora si implementarono controlli frapponendosi tra processo utente e system call, controlli questi sicuramente più pesanti di quelli che saranno utilizzati in Dafne. Per questo motivo la barriera osservata per il processo AngeL rappresenta un limite superiore al degrado di performance che difficilmente verrà avvicinato.

Si osservi infine come un degrado delle performance del sistema può essere di gran lunga tollerato se in gioco c'è un innalzamento del livello di sicurezza dei nostri sistemi.

## Ringraziamenti

Doverosi ringraziamenti vanno ai ragazzi di `sikurezza.org` che ospita, oltre al progetto AngeL e lo stesso progetto dafne, una delle più importanti mailing list riguardanti la sicurezza informatica rivolta soprattutto al panorama italiano.

Un grazie a tutte le persone che hanno accolto la mia richiesta di review della documentazione e che hanno partecipato con commenti e suggerimenti a questa prima fase di stesura del progetto.

## Riferimenti bibliografici

- [1] Paolo Perego, Aldo Scaccabarozzi, "*AngeL - The power to protect*", <http://www.sikurezza.org/angel>, 2001
- [2] Linus Torvalds *The Linux Kernel website*, <http://www.kernel.org>
- [3] Snort Development Team *Snort, The Open Source Network Intrusion Detection System*, <http://www.snort.org>