Virt&l-Comm.4.2013.9

UNIVERSITÀ DEGLI STUDI DI CAMERINO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Laurea Specialistica in Informatica

Dipartimento di Matematica e Informatica

REALIZZAZIONE DI UN SISTEMA DI INTRUSION PREVENTION SYSTEM (IPS): FIREWALL, ROUTING E GSM

Tesi di Laurea Sperimentale
in
Reti di Elaboratori

Laureando:
Carlo MANUALI

Relatore:
Dott. Fausto MARCANTONI

Anno Accademico 2005 / 2006

ISSN: 2279-8773

Ringraziamenti

Si ringrazia per la disponibilità dimostrata tutto il Dipartimento di Matematica ed Informatica, le Segreterie Studenti ed in particolare tutti i docenti di questo corso di Laurea Specialistica che hanno sempre dimostrato massima pazienza e professionalità nei miei confronti.

Inoltre, un ringraziamento particolare va a tutti coloro che hanno collaborato alla produzione di questo lavoro di tesi: amici competenti, colleghi del settore e quindi in special modo al mio relatore, Dott. Fausto Marcantoni.

Ringrazio poi Claudia e Martina, che per motivi del tutto diversi ancora mi sopportano, ed infine mia madre Liana, perché senza di lei ogni cosa sarebbe stata semplicemente impossibile.

a mio Padre.

ISSN: 2279-8773

Indice

ISSN: 2279-8773

ISSN: 2279-8773

ISSN: 2279-8773

Introduzione

”Too many people are thinking of security instead of opportunity. They seem more afraid of life than death.”

-- James Francis Byrnes (1879-1972)

Questa tesi presenta un’architettura originale per la realizzazione di un sistema di Intrusion Prevention basato sulla rilevazione di anomalie nel traffico di una rete informatica.

Viene analizzato innanzitutto il problema della sicurezza dell’informazione e degli attacchi contro i sistemi informatici.

A partire da questa analisi, viene introdotto il tema della realizzazione degli Intrusion Prevention System (IPS) e viene studiato lo stato dell’arte di questi sistemi.

Vengono mostrate le relative problematiche che essi

ISSN: 2279-8773

devono affrontare; sulla base di tali osservazioni viene proposta un'architettura innovativa per la realizzazione di un sistema di IPS basato su azioni reattive e dinamiche a fronte di particolari attacchi o condizioni critiche.

Di tale architettura viene presentata una visione d'insieme e vengono studiate le reazioni possibili che prevedono, tra l'altro, l'interazione con un firewall perimetrale, un router di default per le sottoreti protette ed un sistema di alerting e di notifica mediante Global System for Mobile Communications (GSM).

L'obiettivo di questo lavoro di tesi è quindi quello di studiare concretamente la fattibilità della realizzazione e dell'utilizzo di tale sistema di Intrusion Prevention.

I risultati ottenuti dimostrano l'applicabilità di questo approccio originale per individuare, ma altresì per prevenire, gli attacchi informatici.

Inoltre, analizzando gli articoli ed i progetti scientifici prodotti sul tema, è possibile notare che, aldilà della fiducia quasi unanimemente espressa dagli esperti del settore circa la possibilità di realizzare sistemi di prevenzione di

ISSN: 2279-8773

questo tipo, nella pratica ben pochi di tali esperimenti sono stati realizzati concretamente.

Per questo motivo abbiamo deciso di presentare un modello completo di IPS che interagisca con altri apparati presenti in rete in materia di sicurezza quali firewall e router, ma che allo stesso tempo utilizzi diverse tecnologie, come ad esempio GSM piuttosto che la consueta posta elettronica, per tutte quelle operazioni concernenti la notifica degli eventi e degli allarmi.

Questo approccio sperimentale ci ha di fatto obbligato ad analizzare approfonditamente il tema e ci ha portato a proporre ed a realizzare il presente lavoro di tesi organizzandolo come di seguito.

Inizieremo nel primo capitolo con un'analisi generale del problema della sicurezza dell'informazione e delle minacce che oggi minano alla base tale sicurezza.

Prendendo spunto da questa analisi, nel secondo capitolo effettueremo una disamina approfondita del problema dell'Intrusion Detection e Prevention attraverso una descrizione per quanto possibile completa dello stato

ISSN: 2279-8773

dell'arte, cercando di evitare un elenco di articoli e di prodotti, ma piuttosto proponendo una tassonomia dei differenti approcci e dei loro immancabili problemi.

Proporremo poi nel terzo capitolo il software utilizzato e l'architettura del sistema sulla quale la sperimentazione effettuata si poggia ed infine, nel quarto ed ultimo capitolo, la realizzazione di tale progetto.

In conclusione cercheremo di valutare il potenziale di questa soluzione, offrendo, si spera, degli interessanti spunti per diverse elaborazioni e ricerche successive.

ISSN: 2279-8773

Capitolo 1 -

Sicurezza dell'Informazione

Non è certo necessario dimostrare che nel nostro mondo una parte sempre maggiore di ciò che facciamo, di ciò che gestiamo, o che trasmettiamo, viene realizzata, controllata, o coadiuvata dall'utilizzo di sistemi informatici.

Per questa ragione è particolarmente importante capire quanto possiamo fidarci di questi sistemi.

Questo problema diviene sempre più critico, mano a mano che il valore delle informazioni scambiate in forma digitale cresce.

La sicurezza dell'informazione ha attraversato un lungo periodo di gestazione in cui veniva considerata un'arte più che una scienza, da apprendere tramite l'esperienza e non tramite lo studio.

Solo pochi studi pionieristici stendevano le basi dei metodi formali che negli ultimi anni sono stati sviluppati.

Questo, unito all'intrinseca difficoltà della materia, spiega il

ISSN: 2279-8773

motivo per cui gran parte degli approcci al tema della sicurezza dell'informazione sono basati su esperienza e conoscenza euristica più che su tecniche raffinate.

L'unico settore della sicurezza che vanta una lunga tradizione nell'uso dei metodi formali è il campo della crittografia, che, non sorprendentemente, è molto più avanzato e stabile degli altri.

Se questo metodo pragmatico di procedere ha funzionato per anni, lo sviluppo esplosivo dei sistemi collegati in rete e di dispositivi con potenza e versatilità francamente inimmaginabili solo pochi anni fa ha reso necessario, quasi urgente, lo sviluppo di un'ingegneria della sicurezza informatica, dove il termine "ingegneria" va inteso nel suo senso tradizionale di scienza applicata, con definizione formale di concetti, metodi e tecniche conosciute, esplorate da un'ampia letteratura scientifica, e globalmente diffuse.

1.1 Il paradigma CID

Sia nella letteratura accademica che nella pratica, gli obiettivi della sicurezza dell'informazione sono

ISSN: 2279-8773

normalmente descritti in termini di confidenzialità, integrità e disponibilità dell'informazione stessa (il paradigma CID o, usando i termini inglesi - Confidentiality, Integrity and Availability - CIA paradigm).

La confidenzialità dell'informazione può essere definita in vari modi, ma è sostanzialmente la capacità di un sistema di rendere le proprie risorse accessibili soltanto a coloro che sono autorizzati ad accedervi.

In modo formale, possiamo dire che la confidenzialità costruisce una relazione tra ogni risorsa del sistema e uno o più soggetti autorizzati all'accesso in lettura.

Per fare un parallelo con il mondo reale, si può pensare ai livelli utilizzati dalla NATO per i documenti "classified": Confidential, Secret e Top Secret.

Ogni persona ha, o non ha, una "clearance" per un determinato livello.

Inoltre, per ogni documento si decide il "need to know", ovvero la necessità per una determinata persona di accedere alle informazioni contenute.

Questo tipo di modello è stato trasportato nell'informatica

ISSN: 2279-8773

mediante i Trusted Computer System Evaluation Criteria (TCSEC) [1].

L'integrità è invece definita come la capacità di un sistema di rendere possibile solo alle persone autorizzate la modifica delle sue risorse e dei suoi dati, e anche a queste solo in modo autorizzato onde garantire la consistenza di questi dati con le funzioni gestite dal sistema stesso.

Anche questa definizione implica la creazione di una relazione tra ogni risorsa ed un insieme di utenti autorizzati; tuttavia ci si rende immediatamente conto della difficoltà di definire, e ancor più di garantire, che ogni utente modifichi in modo corretto soltanto i dati che la sua posizione gli da autorità di modificare.

Si può inoltre notare come la creazione di una relazione tra ogni risorsa e gli utenti che vi hanno accesso è perfettamente equivalente alla costruzione di una relazione tra ogni utente e l'insieme di risorse a cui può accedere.

Sarebbe umanamente più comprensibile utilizzare il secondo formato, ma per ragioni storiche ed implementative i sistemi reali utilizzano controlli d'accesso

ISSN: 2279-8773

che riflettono la prima forma.

È proprio questo semplice problema di prospettive algebriche a creare molti dei problemi di mis-configurazione di cui discuteremo nel seguito.

La disponibilità è in qualche modo un obiettivo in conflitto con gli altri due, in quanto impone che le relazioni precedenti non siano semplicemente dei "solo se", ma dei "se e solo se", ovvero il sistema deve garantire accesso soltanto le persone autorizzate, ma non deve mai negare l'autorizzazione ad un accesso corretto.

Inoltre, l'accesso deve essere consentito tempestivamente, dove la definizione esatta di "tempestivamente" dipende dal dominio del problema.

Esiste in letteratura [2] il concetto di Maximum Waiting Time proprio introdotto a questo proposito.

Spesso vengono utilizzati i termini "exposure" per indicare il fallimento nella condizione di confidenzialità, "compromise"per il fallimento nella relazione di integrità, e "denial of service" per la mancanza di disponibilità, tuttavia su questa terminologia non c'è lo stesso accordo che sulla

ISSN: 2279-8773

precedente [3].

Altri testi definiscono, in aggiunta alle tre condizioni elencate, ulteriori obiettivi quali "privacy" o "identificazione", ma bastano poche e semplici valutazioni per comprendere come, in generale, si tratti di semplici sottocasi di questi obiettivi più generici, oppure di tecniche utili a raggiungere tali obiettivi.

Per esempio la privacy non è altro che la confidenzialità dei dati sensibili dell'utente, mentre il "non ripudio" è la capacità di un sistema di provare oltre ogni dubbio che una transazione è avvenuta.

Esempi di tecniche spesso confuse con i fini della sicurezza sono l'identificazione e l'autenticazione degli utenti, oppure l'accounting, ovvero la possibilità per l'amministratore di sistema di ricostruire a posteriori gli eventi e le responsabilità.

È in ogni caso necessario notare che il concetto di "sicurezza dell'informazione" si estende ben oltre l'informatica in senso ristretto, ovvero l'hardware, il software e le reti di computer, andando a interessarsi del

ISSN: 2279-8773

sistema informativo nel suo complesso, e quindi dell'intero processo di gestione dei dati e delle informazioni messo in atto da un'organizzazione.

Rientrano quindi a pieno titolo nell'ingegneria della sicurezza anche considerazioni di sicurezza ambientale, gestione dell'informazione non digitale o procedure nella gestione delle risorse umane.

In effetti, i filtri della sicurezza dell'informazione operano all'interfaccia tra le persone ed i processi del sistema informativo e devono gestire entrambe le variabili.

Come si può immaginare, si tratta di un processo complesso, costoso e prono a sviste anche gravi.

1.2 L'implementazione tradizionale

L'implementazione tradizionale del paradigma CID, che si può osservare in quasi tutte le applicazioni informatiche e in buona parte dei processi di qualsiasi sistema informativo moderno, è un approccio che viene usualmente riassunto nel motto "Who are you? What can you do?".

Siti web, sistemi operativi pensati per l'uso in rete, sistemi

ISSN: 2279-8773

di accesso remoto, telefoni cellulari, ormai qualsiasi tipo di applicazione accessibile da molti utenti e con esigenze anche minime di sicurezza richiede all'utente di "effettuare un login", ovvero in qualche modo di provare la propria identità, prima di accedere ai servizi del sistema.

L'identità digitale che viene associata all'utente ha privilegi e impostazioni particolari che limitano i servizi e le operazioni ad esso disponibili.

Anche altri servizi ed applicazioni tipicamente usati per garantire la sicurezza in rete, come firewall e virtual private network (VPN), operano con un meccanismo simile, identificando un utente o una macchina in rete, ad esempio per mezzo di un login e di una password, per mezzo dell'indirizzo IP o mediante uno scambio di token crittografici, quindi applicando un insieme di regole alle operazioni che questa persona o macchina può o non può eseguire.

Per effettuare questa associazione esistono diversi tipi di paradigmi, divisi in due macro-categorie: paradigmi Discretionary Access Control (DAC), e paradigmi Mandatory

ISSN: 2279-8773

Access Control (MAC), con una comparsa marginale di paradigmi definiti RBAC, acronimo di Role Based Access Control.

Nel caso dei DAC, la regola di associazione è semplice: ogni oggetto ha un proprietario.

Il proprietario può fare ciò che vuole dell'oggetto, delegando diritti a chi desidera all'interno del sistema su di esso, fino ad arrivare alla possibilità di trasferire ad un altro utente l'ownership dell'oggetto.

Esiste in generale un amministratore di sistema che ha l'ownership degli oggetti di sistema, oltre a poter violare le restrizioni d'accesso.

I sistemi DAC sono sicuramente quelli a cui siamo più abituati: i meccanismi dei permessi UNIX e le ACL degli oggetti Windows NT sono infatti sistemi DAC.

In un sistema MAC gli oggetti hanno un loro livello di segretezza (potrebbero essere, in generale, più livelli in diverse categorie, ma per comprendere il meccanismo basta pensare a un singolo livello omnicomprensivo).

Gli utenti hanno un loro livello di accesso: esiste quindi una

ISSN: 2279-8773

figura che nei sistemi normali non è prevista, ovvero quella del "security officer", che è l'unico a poter amministrare i livelli d'accesso.

In generale le regole che vengono usate per stabilire i permessi d'accesso sono il "no read up", ovvero l'utente non può leggere informazioni con un livello d'accesso superiore al suo, e il "no write down", ovvero l'utente non può scrivere informazioni verso un livello d'accesso inferiore al suo.

Il modello MAC forse più famoso, quello di Bell-LaPadula [4], descrive le regole d'accesso mediante questi due principi:

- *security rule*, un utente non può leggere informazioni il cui livello d'accesso sia più elevato del suo;
- *property*, un utente non può spostare informazioni da un livello di sicurezza a un livello di sicurezza più basso, ma può spostarle ad uno più alto; questo coincidentalmente include la regola di "no write down".

Per necessarie ragioni di praticità, il sistema di Bell-LaPadula, nella sua descrizione originaria, prevedeva la

ISSN: 2279-8773

possibilità di utilizzare un sistema DAC a permessi per gli oggetti "non confidenziali", e la possibilità di definire dei "trusted subjects" (ad esempio lo spooler della stampante) in grado di violare le regole d'accesso per svolgere compiti di sistema.

I sistemi di tipo RBAC, molto studiati negli ultimi anni, tendono a superare entrambi questi concetti definendo il paradigma dei "ruoli".

I diritti d'accesso agli oggetti sono definiti in base a dei ruoli, come ad esempio l'accesso al database "paghe" è consentito al ruolo "direzione_risorse_umane", e uno o più ruoli sono associati agli utenti in base alle loro esigenze lavorative.

Il sistema sembrerebbe essere rigoroso ed allo stesso tempo flessibile: difatti ricerche e sperimentazioni sono ad oggi in corso.

Quasi ogni sistema di sicurezza tradizionale è un'implementazione più o meno evoluta del paradigma base di identificazione e gestione dei privilegi: il sistema chiede ad un utente, che può essere una persona o ancora

ISSN: 2279-8773

un altro sistema, di identificarsi e usa questa identità per attivare un insieme di regole operative.

Ad esempio, i sistemi di controllo d'accesso da remoto funzionano tutti in base a questo schema; ma anche sistemi apparentemente diversi, come ad esempio i firewall, controllano il traffico in ingresso e in uscita da una rete, "identificandone" la provenienza ed applicando opportune politiche di filtraggio.

Questo approccio ha successo perché è quasi del tutto ortogonale al problema: non importa infatti quale sia nello specifico il protocollo o il servizio che si offre, ma uno schema di autenticazione e di privilegi può esservi imposto comunque senza troppa fatica; in generale, anche troppi software e diversi protocolli vengono dapprima fatti funzionare sotto l'assunzione che chiunque vi possa avere accesso illimitato, dopodichè si provano a creare dei controlli successivi che di fatto vadano ad aggiungere uno strato di sicurezza da imporre sul sistema.

ISSN: 2279-8773

1.3 La tassonomia delle minacce

Cercheremo di dare una tassonomia ed una nomenclatura per gli attacchi ai sistemi informatici e per i possibili esecutori di questi attacchi.

Si premette che questa non intende essere in alcun modo una guida esaustiva a tutti i tipi di attacchi, un compito che si rivelerebbe improbo, visto il numero quotidiano di innovazioni in questo settore, ma piuttosto cercherà di delineare una serie di direttrici tipiche d'attacco e di tecniche "esemplari" universalmente note.

Vi sono molte ricerche che propongono una tassonomia di attacchi, ad esempio [5], ma quasi tutte sono più o meno incomplete e contraddittorie.

Non esistendo un vero consenso sull'argomento, ci limitiamo quindi a proporre alcuni concetti di base che serviranno nelle discussioni che seguono.

Uno dei più interessanti ed informati tentativi di descrivere un incidente informatico e dare una tassonomia delle sue componenti è stato fatto dal CERT/CC [6].

In particolare in fig.1 viene descritto l'incidente informatico

ISSN: 2279-8773

come un processo con componenti multipli: la ricerca del CERT approfondisce quindi ognuno degli elementi dello schema, ma nel seguito ci si distaccherà da esso integrandolo o sostituendolo con il nostro punto di vista e le nostre esperienze [7], e coerentemente con la nostra finalità espositiva.

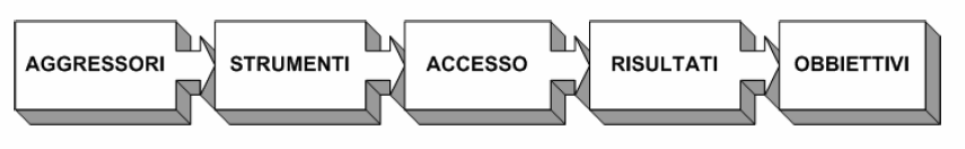

Fig. 1 - L'intrusione come processo.

"Conoscere l'altro, e se stessi – cento battaglie, nessun rischio. Non conoscere l'altro, e conoscere se stessi – a volte vittoria, a volte sconfitta. Non conoscere né l'altro né se stessi – ogni battaglia è un rischio certo" [8].

Le parole del saggio Sun Tzu, maestro cinese dell'arte della guerra, ci devono fare riflettere.

Per capire l'andamento di qualsiasi scontro bisogna conoscere le due parti in gioco, e nel campo di battaglia della sicurezza vi sono aggressori e aggrediti, o attaccanti e

ISSN: 2279-8773

difensori.

Ponendoci nell'ottica dei difensori, conoscere "noi stessi" significa conoscere i sistemi che utilizziamo ma soprattutto conoscere cosa dobbiamo proteggere.

Conoscere il nemico significa essere consci delle sue capacità e dei suoi obiettivi.

Per questo ci pare utile affrontare in modo coordinato l'inizio e la fine del processo dell'incidente informatico, ovvero l'aggressore e i suoi obiettivi.

La prima nota da fare è che, eccettuati rari casi, i sistemi informativi non vengono aggrediti da altri sistemi, ma da persone.

Ciò implica che l'analisi delle loro motivazioni e delle loro metodologie sia una disciplina delle scienze sociali e della psicologia più che dell'informatica.

Tuttavia, come abbiamo già notato, la disciplina della sicurezza dell'informazione è una di quelle discipline che si situano all'interfaccia tra l'uomo e il sistema informativo, affrontando una miscela unica ed interdisciplinare di problemi paragonabili, ad esempio, a quelli riscontrati nello

ISSN: 2279-8773

studio delle Human Computer Interfaces (HCI).

Per cominciare, possiamo fare una macrodistinzione tra minacce esterne e minacce interne.

Come abbiamo già detto infatti, una gran parte degli attacchi provengono da dentro le mura di una qualsiasi organizzazione.

Già definire i "confini" di una singola organizzazione diviene oggi difficile: concetti come le "extranet" che estendono il sistema informativo mediante Internet fino ad interfacciarlo con quello dei partner, dei clienti e dei fornitori, rendono molto labile il confine tra ciò che è interno ad un'impresa e ciò che è esterno.

Un aggressore interno ha tutti i vantaggi, in quanto molto spesso opera a scatola aperta: esso infatti sa già quali possono essere i punti deboli e come è strutturata l'architettura dei sistemi interni, mentre dall'esterno le stesse cose vanno scoperte poco alla volta.

Tra gli aggressori interni possiamo individuare due categorie principali:

- *disgruntled employees*, che sono persone che stanno

ISSN: 2279-8773

coscientemente aggredendo l'organizzazione, perché hanno qualche conto in sospeso, o perché sono stati corrotti, o ancora peggio che si sono fatti assumere appositamente per avere un'opportunità di violarne la sicurezza;

- *trasgressori ingenui*, che semplicemente cercano di abusare in qualche modo delle risorse loro concesse, per esempio violando le linee guida aziendali sull'uso responsabile di Internet, sul software da installare sulle postazioni di lavoro aziendali o simili.

  Sono da controllare con attenzione perché nel loro tentativo di violare le difese aziendali potrebbero involontariamente fornire una porta d'accesso ad altri intrusi ben più pericolosi.

Tra gli aggressori esterni possiamo fare due tipi di distinzioni; la prima è una distinzione a livello di capacità, e distinguiamo:

- *esperti*, ovvero persone dotate di conoscenze tecniche approfondite, capaci di elaborare attacchi innovativi o di "inventare" procedimenti nuovi; molto spesso sono più

ISSN: 2279-8773

esperti dei sistemi che stanno attaccando di quanto non siano le persone incaricate di difenderli.

Se volessimo tracciarne un quadro psicologico li definiremmo come calcolatori, freddi, sul limite del paranoico, desiderosi di avere sempre sotto controllo la situazione;

- *script-kiddies*, ovvero persone prive di reali conoscenze tecniche, che studiano ed utilizzano "exploit", attacchi preconfezionati rilasciati da altri esperti; il loro quadro psicologico è ben più agitato, emotivo, possono lasciarsi prendere dalla paura e quindi agire irrazionalmente.

Tra le due categorie vi sono ovviamente infinite gradazioni intermedie; una seconda distinzione dei nemici si può fare anche secondo i loro fini:

- *nemici con un obiettivo preciso*, infatti sul sistema informativo sono contenuti dati che vogliono rubare, oppure vogliono informazioni su una singola persona, piuttosto che modificare o eliminare qualche informazione scomoda, o in ogni caso mirano ad un obiettivo specifico ed a lasciare quante meno tracce

ISSN: 2279-8773

possibili circa la propria presenza;

- *nemici che vogliono fare soltanto un danno*, per esempio per rabbia, perché sono stati pagati per farlo, o perché sono l'equivalente telematico dei vandali;
- *persone che stanno giocando*, che esplorano sistemi che non gli appartengono e se li vedono vulnerabili provano a bucarli come "sfida" alle proprie capacità.

  Non va assolutamente sottovalutata la potenzialità di questi ultimi aggressori "ludici", e quindi nemmeno la loro pericolosità.

  Infatti, per quanto spesso non vogliano fare alcun danno, o addirittura sperino di "dare una mano" agli amministratori delle macchine che attaccano, un errore può sempre succedere.

Dobbiamo anche valutare quali siano i possibili bersagli di un aggressore, ovvero secondo degli obiettivi precisi:

- *i dati*, l'aggressore cerca di violare la confidenzialità delle informazioni che il sistema contiene, oppure cerca di modificarle abusivamente.

  I dati possono essere copiati per essere venduti, oppure

ISSN: 2279-8773

dare vita a un rapimento con riscatto: ad esempio l'attaccante può ottenere abusivamente i segreti industriali di una vittima e minacciare di renderli pubblici se non gli verrà versato un cospicuo compenso;

- *i sistemi*, l'aggressore mira alle risorse telematiche in sé più che al loro contenuto.

  Ad esempio, moltissimi script-kiddie cercano macchine facili da penetrare che possano essere usate come "deposito" di materiale pornografico o piratato, o che possano essere abusate per fare spam, piuttosto che per disporre di connessioni permanenti per qualsivoglia altra ragione;

Le finalità ultime di una persona possono essere, come abbiamo visto, moltissime: dal guadagno diretto al guadagno derivante dalla vendita di informazioni, dalla rappresaglia al divertimento; per cui non ci avventureremo oltremodo su questo campo che è proprio della criminologia e su cui in realtà è tuttora in corso una fervida attività di studio, anche da parte di ricercatori italiani [9].

Il cuore dell'attacco informatico, come abbiamo detto, è

ISSN: 2279-8773

ottenere in qualche modo un accesso non desiderato al sistema: ottenere la possibilità di leggere dati confidenziali, ottenere la possibilità di modificare dati protetti, o ancora ottenere la possibilità di intralciare l'accesso legittimo ai dati da parte di altri utenti.

Pertanto il punto focale sono i dati e le risorse gestite dal nostro sistema e le applicazioni o processi che il sistema svolge.

L'aggressore vuole accedere in modo non autorizzato o comunque non previsto a questi dati e per farlo sfrutta necessariamente una qualche vulnerabilità.

Ecco lo schema del CERT, da noi integrato in fig.2:

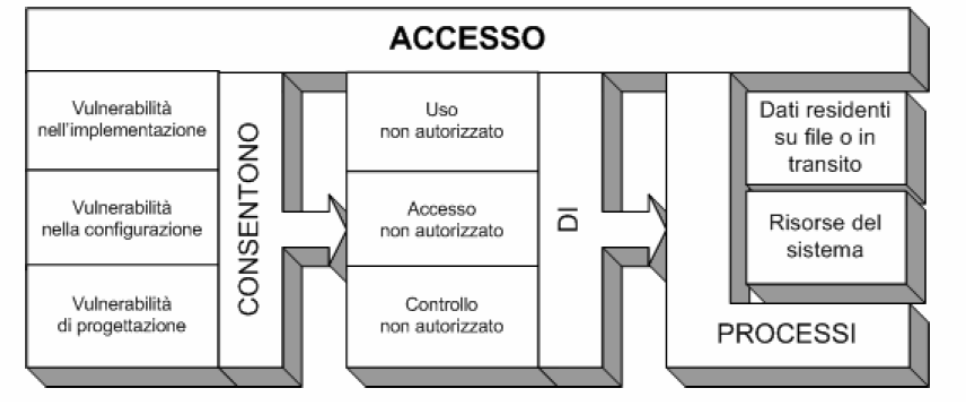

Fig. 2 - Schema dell'accesso non autorizzato ad un sistema informatico.

ISSN: 2279-8773

Dal momento che lo scopo è quello di violare in qualche modo le restrizioni di confidenzialità, integrità o disponibilità imposte sui dati, l'aggressore deve, necessariamente, attivare, controllare, oppure fare fallire uno o più dei processi del sistema informatico, in un modo non previsto nel suo design originario.

In poche parole, l'uso o l'accesso non autorizzato ai processi del sistema consentono all'intruso di violare la sicurezza dei dati.

Ma come è possibile che avvenga ciò?

Esistono tre tipologie fondamentali di vulnerabilità in un sistema informatico:

• *vulnerabilità nell'implementazione dei programmi*, che consentono all'aggressore di far comportare il programma in maniera diversa da quella attesa.

Non possiamo riportarne un elenco completo, per il quale rimandiamo a testi specifici [10], ma le più comuni sono:

- *errori nella validazione dell'input*, ovvero nel filtraggio dei dati che arrivano da fonti inaffidabili, ad esempio gli utenti.

ISSN: 2279-8773

Un esempio concreto e molto comune sono le pagine web che gestiscono form collegate all'esecuzione di query SQL sui database; a volte gli autori dimenticano di controllare che non vengano inseriti caratteri speciali nei campi, rendendo possibile all'utente "modificare" non solo i parametri, ma il codice SQL in sé;

- *problematiche di buffer overflow*, ovvero la possibilità che un insieme di valori troppo lungo venga inserito in un buffer di lunghezza prefissata, andando a influire sullo stack [11], alterando il flusso di esecuzione del programma, o sullo heap del programma, corrompendone le variabili;

- *problemi nelle chiamate e comunicazioni tra processi e nella gestione della memoria*, ad esempio, passaggi in chiaro di dati molto sensibili che possono essere intercettati da qualche curioso;

- *problemi di gestione dei privilegi di esecuzione*, in particolare per i demoni di rete che hanno bisogno dei privilegi di root in alcuni punti della loro esecuzione;

ISSN: 2279-8773

- *attacchi relativi al timing di determinate operazioni critiche*, ovvero i cosiddetti "race condition attacks";
- *problemi relativi all'implementazione degli algoritmi crittografici*, e quindi di utilizzo di fonti di casualità di bassa qualità;

- *vulnerabilità nella configurazione*, molti programmi per essere considerati sicuri debbono essere opportunamente modificati, configurandoli ed adattandoli alle situazioni specifiche in cui vengono inseriti.

  Spesso quindi è proprio nella fase di configurazione che amministratori inesperti compiono gravi leggerezze, che rendono di fatto inutili tutte le contromisure di sicurezza. Per esempio, la condivisione di unità disco in ambiente Windows senza alcuna forma di autenticazione è da anni costantemente nella lista delle "20 vulnerabilità più diffuse" gestita dal SANS institute [12];
- *vulnerabilità nella progettazione*, di una determinata applicazione, sistema, o protocollo, che la rendano insicura per come è stata creata.

ISSN: 2279-8773

Esempio tristemente noto e recente è il protocollo crittografico WEP, Wired Equivalent Privacy, progettato per rendere sicure le comunicazioni sulle reti Wireless LAN realizzate secondo lo standard IEEE 802.11b; un errore di design rende tale protocollo estremamente debole e facile da decodificare.

Abbiamo volutamente ordinato tali classi in ordine crescente di "livello concettuale", e di difficoltà a correggere l'errore una volta individuato: un buffer overflow può essere corretto mediante una patch; un errore di configurazione, una volta scoperto, può essere sistemato; un algoritmo come WEP è irrecuperabile, e può quindi solo essere abbandonato in favore di un altro.

Le abbiamo anche scelte in modo che fossero del tutto ortogonali: una corretta implementazione di un progetto errato o una cattiva configurazione di un servizio concettualmente ben studiato e appropriatamente implementato, sono tutti casi egualmente possibili.

All'interno di queste macroclassi, esistono una quantità enorme di vulnerabilità più o meno note; esistono infatti

ISSN: 2279-8773

intere classi di weakness che si applicano in maniera trasversale a decine di software diversi, a volte aventi funzioni totalmente differenti.

Nonostante esistano copiosi volumi di "best practices" su come scrivere un'applicazione sicura, su come progettare in modo sicuro sistemi e protocolli, le stesse, vecchie vulnerabilità e gli stessi problemi noti, continuano a ricomparire, rendendo tristemente evidente che anche se fosse possibile enumerare tutti i principi della programmazione e della progettazione sicura, nondimeno il fattore umano interverrebbe a minare qualsiasi sforzo.

Per sfruttare una vulnerabilità al fine di far agire un programma in modo erroneo e guadagnare così l'agognato accesso ai dati, l'intruso interagirà con il sistema seguendo una procedura che viene comunemente denominata "exploit".

A volte, per gli attacchi svolti mediante dei particolari tool, script o programmi, si usa il termine "exploit" per indicare proprio questi ultimi, chiamando invece "exploit techniques" le sequenze prolungate di azioni tese a violare

ISSN: 2279-8773

un sistema.

In generale si pone una netta distinzione tra attacchi di tipo remoto e attacchi di tipo locale: non si intende distinguere, banalmente, che l'attaccante stia seduto o meno alla tastiera della macchina attaccata.

Un attacco da remoto è piuttosto un attacco per il quale l'aggressore non parte da un accesso a livello comandi alla macchina mentre un attacco di tipo locale è perpetrato a partire dall'accesso a livello comandi, che tuttavia in un sistema operativo di rete può essere usualmente consentito anche da remoto, per esempio tramite telnet o SSH.

In generale questi due attacchi corrispondono a due fasi successive, denominate break-in e privilege escalation.

Break-in è l'attacco con cui un intruso partendo da remoto ottiene un accesso sufficiente per portare attacchi di tipo locale.

Il termine "escalation" fa ovviamente riferimento alla salita da un livello da "ospite" a quello di utente effettivo del sistema, poi a quello di amministratore, ad esempio l'utente super-user dei sistemi UNIX-like.

ISSN: 2279-8773

Esistono però attacchi che da remoto consentono a un intruso di ottenere direttamente i privilegi amministrativi, e vengono significativamente categorizzati come "remote root exploits": ovviamente questo tipo di attacchi sono tra i più pericolosi.

Bisogna altresì notare che alcuni attacchi, per esempio molti denial-of-service, non cercano di guadagnare accesso al sistema ma semplicemente di negarlo ad altri utenti.

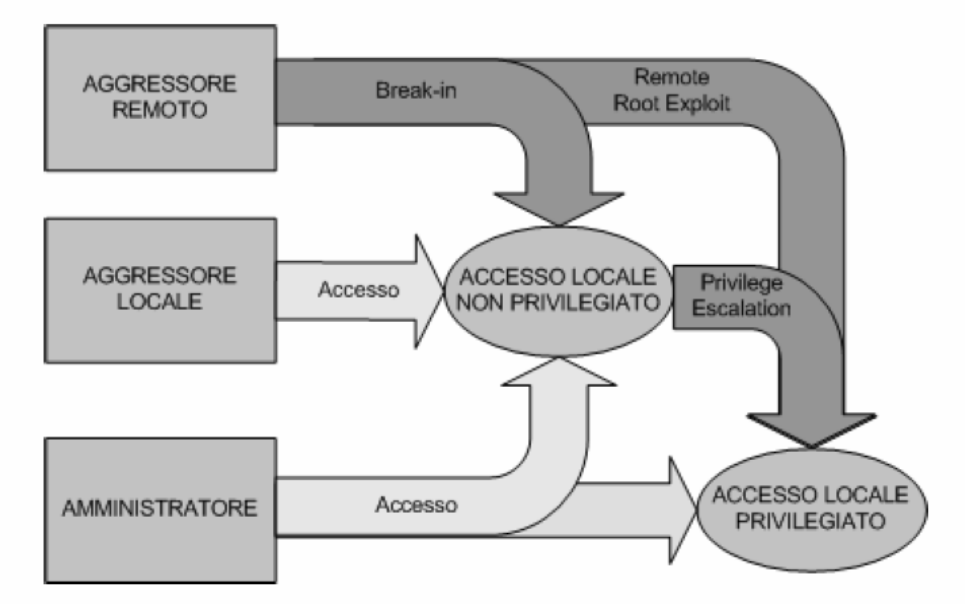

Fig. 3 - I cammini tipici per l'accesso ad un sistema informatico: in colore scuro quelli illeciti.

ISSN: 2279-8773

Per questo molti di questi attacchi possono essere eseguiti da remoto senza bisogno di un vero e proprio break-in.

Per ottenere questo tipo di accessi "illeciti" mostrati in fig.3, gli aggressori usano una vasta serie di strumenti, per i quali non è possibile certo presentarne un elenco esaustivo.

Tuttavia proviamo ad elencarne le tipologie principali:

- *strumenti di packet forgery*, vasta categoria di programmi, sia generali, ovvero che consentono di costruire pacchetti del tipo desiderato dall'utente, sia molto specifici, ovvero che creano pacchetti strutturati appositamente per sfruttare una specifica vulnerabilità.

  Molti "remote exploit tools" ricadono in quest'ultima categoria;
- *port scanner e vulnerability scanner*, strumenti che inviano ad una o più macchine pacchetti TCP/IP per cercare di scoprire quali servizi sono disponibili.

  I vulnerability scanner aggiungono a questa funzione una base di conoscenze per determinare se tali servizi siano vulnerabili ad attacchi noti, sia mediante funzioni

ISSN: 2279-8773

di raccolta di informazioni, che mediante veri e propri tentativi di attacco;

- *normali client d'accesso*, non sono poche le vulnerabilità che, per essere sfruttate, non richiedono nient'altro che un normale browser, un client telnet, o qualche altro tipo di client e un pò di conoscenze da parte dell'aggressore.

  Ricadono in questa categoria, per esempio, gran parte delle vulnerabilità dovute a cattiva configurazione.

  A volte si usano dei client modificati: per esempio un client FTP da linea di comando, insieme ad uno script, può essere utilizzato per fare molteplici tentativi di accesso tentando di "indovinare" una password;
- *programmi da eseguire*, o far eseguire con l'inganno su un host remoto per ottenere gli effetti più vari.

  Un noto detto della sicurezza informatica vuole che "una volta che qualcuno sia riuscito, con l'inganno o mediante qualche tecnica, ad eseguire o a farti eseguire un programma di sua scelta sulla tua macchina, quella non sia più la tua macchina";

ISSN: 2279-8773

- *agenti software autonomi*, con varie funzioni, sono dei programmi simili ai precedenti ma sono in grado di attuare da soli un processo di compromissione del sistema;
- *artefatti fisici*, un tempo erano molto di moda i "wiretap", piccoli morsetti applicati sui cavi di una rete di telecomunicazioni per intercettare e inserire traffico abusivamente, al punto che molte canaline di rete sicure venivano integrate con impianti ad aria compressa per rilevare eventuali forature.

  Caduta in disuso negli anni '90, la pratica sta rifiorendo con le reti wireless, in cui l'uso di apparecchiature di intercettazione dei segnali anche molto semplici consente risultati sorprendentemente efficaci.

Strumenti e accessi si combinano in un numero sorprendente di scenari di intrusione possibili, di cui elenchiamo le differenze più comuni ed i relativi risultati.

1.4 I tipi di attacchi

Distinguiamo i tipi di attacchi in attacchi da remoto,

ISSN: 2279-8773

attacchi locali ed attacchi ibridi.

Per i primi potremmo effettuare la seguente classificazione:

- *attacchi a demoni e servizi di rete*, ovvero il tipo di exploit forse più conosciuto, al punto tale da essere spesso l'unico preso in considerazione.

  L'idea alla base di questi attacchi è che, in assenza di un accesso diretto al livello comandi del sistema, l'aggressore cerchi di fare in modo che un processo già in esecuzione, spesso con elevati privilegi, esegua al posto suo delle operazioni sul sistema:

  - *mediante l'invio di pacchetti artefatti*, spesso i progettisti di applicazioni di rete dimenticano che è possibile per chiunque forgiare ed iniettare pacchetti all'interno della rete Internet, e molti servizi si sono di fatto dimostrati vulnerabili a particolari pacchetti malformati;
  - *mediante l'interazione con l'interfaccia pubblica*, molti attacchi vengono portati interagendo con l'applicazione, demone o servizio, esattamente come qualsiasi utente normale farebbe, in realtà inviando

ISSN: 2279-8773

dati o comandi malformati che poi provocano effetti indesiderabili;

Possiamo ricordare i bug Unicode di Internet Information Server (IIS) di Microsoft, il cui meccanismo di exploit era una richiesta HTTP malformata, piuttosto che le vulnerabilità delle applicazioni web-based in generale;

- *grazie a mis-configurazioni*, la più tipica delle vulnerabilità da remoto è una mis-configurazione o un'errata concezione della sicurezza, come ad esempio pagine web di amministrazione lasciate on-line senza password o directory FTP aperte a tutti in l'upload;
- *mediante tecniche di bruteforcing*, l'aggressore cerca, per tentativi, di individuare i codici di accesso di un servizio, nella speranza di trovarne di deboli o corti; contrariamente a quanto si crede, è un attacco relativamente poco usato perché facilmente individuabile;

• *attacchi all'infrastruttura*, ovvero attacchi che non si

ISSN: 2279-8773

concentrano sul sistema aggredito, ma che cercano di sfruttare alcune debolezze dell'infrastruttura di rete che lo circonda, per esempio:

- *sniffing*, dove i dati su Internet viaggiano in chiaro e vengono trasmessi da un punto all'altro rimbalzando per una serie di sistemi di cui non conosciamo l'affidabilità; non solo, ma su una rete locale Ethernet, tranne in alcuni casi, tutte le workstation sullo stesso segmento di rete vedono passare tutto il traffico destinato anche alle altre macchine.

  La pratica dello sniffing consiste proprio nell'analizzare il traffico abusivamente intercettato, in generale alla ricerca dei codici d'accesso per le macchine;
- *hijacking*, riferito alle connessioni TCP è un tipo d'attacco tanto pericoloso quanto difficile da realizzare, che consente a un aggressore di "rilevare" la connessione stabilita tra due macchine, e continuare ad utilizzarla.

  Per fare un esempio, un intruso potrebbe attendere

ISSN: 2279-8773

che A abbia stabilito una connessione telnet con B per poi "inserirsi" al posto di A e sfruttare il collegamento per fare ciò che vuole su B;

- *man-in-the-middle*, un attacco in cui l'intruso, C, riesce a interporsi in maniera perfetta tra due sistemi, A e B, che stanno comunicando via rete.

  In pratica A parla con C, illudendosi di parlare con B; B pure parla con C, interpretando le sue risposte come provenienti da A;

- *spoofing*, pratica che prende le mosse dall'osservazione che nel protocollo IP non c'è un meccanismo di autenticazione del mittente; l'indirizzo IP sorgente può pertanto venire tranquillamente falsificato.

  A livello di TCP sono previsti alcuni meccanismi, come la presenza di un numero pseudocasuale di sincronizzazione, che rendono più difficile questo attacco "alla cieca"; di fatto però se l'aggressore vede tutti i pacchetti mentre passano, ad esempio mediante sniffing, la difficoltà non è insormontabile;

ISSN: 2279-8773

- *denial-of-service da remoto*, attacchi spesso meno complessi dei precedenti, in quanto in questo caso l'obiettivo dell'aggressore è "semplicemente" impedire a chiunque altro di utilizzare il servizio aggredito, tramite:
  - *saturazione delle risorse*, sia del particolare servizio, che dell'intero sistema.

    Un esempio di attacco di questa classe può essere il SYN-flood, ovvero un flusso continuo di richieste di connession (SYN) che non vengono confermate con il terzo ACK dell'handshake a tre vie.

    La macchina bersaglio terrà traccia di ognuna di queste connessioni nell'inutile attesa dell'ACK di risposta, e prima o poi esaurirà la sua capacità di accettare nuove connessioni;
  - *utilizzo di pacchetti artefatti*, come per il caso dell'uso di pacchetti malformati per attacchi finalizzati a ottenere un accesso, determinati pacchetti "strani" possono invece ottenere l'effetto di far bloccare un servizio o un'applicazione, ottenendo un denial-of-service;

ISSN: 2279-8773

- *utilizzo dell'interfaccia pubblica del demone*, facendo richieste impossibili, malformate, o semplicemente subissando di richieste molto lunghe il servizio che si vuole bloccare, è quindi possibile impedire l'accesso agli utenti legittimi.

Distinguiamo invece per gli attacchi locali:

- *mediante l'esecuzione di comandi e script*, dall'interno di un computer ed utilizzando i comandi disponibili agli utenti si possono compiere un gran numero di possibili attacchi, cercando di ottenere privilegi "migliori", come l'accesso in lettura a tutti i file, oppure i privilegi di amministratore del sistema;
- *mediante la compilazione e l'esecuzione di exploit sull'host vittima*, esistono alcuni bug di servizi locali o del sistema operativo che possono essere sfruttati mediante l'esecuzione locale di programmi di exploit;
- *denial-of-service da locale*, lanciare un denial-of-service per un utente locale è ancora più facile: il sistema si aspetta, in genere, di dover eseguire le richieste di chi accede dall'interfaccia comandi;

ISSN: 2279-8773

- *saturazione delle risorse*, un utente locale, se non è stato limitato in qualche modo, può lanciare in esecuzione un insieme numeroso di processi, per esempio scrivendo un piccolo programma C che lancia due "copie" di se stesso e la cui crescita esponenziale metterà in breve in ginocchio il sistema;
- *altri tipi di interazione con il sistema*, in quanto quasi ogni sistema ha delle attività, per esempio di manutenzione programmata, in grado di rallentare o bloccare l'accesso ai servizi.

  Inoltre, un intruso con i privilegi di amministrazione può "spegnere" qualsiasi servizio gli interessi disattivare.

Infine un'ultima classificazione per gli attacchi ibridi riguarda:

- *utilizzo di agenti autonomi*, questi programmi agiscono da soli, in modo indipendente da chi li ha creati e lanciati, per guadagnare accesso ad altre macchine ed eseguirvi operazioni non autorizzate; l'esempio più tipico sono i virus e i worm;
- *virus*, ovvero "frammenti di codice eseguibile che

ISSN: 2279-8773

quando vengono lanciati copiano sé stessi all'interno di altri programmi eseguibili.

Di fatto quando questi ultimi vengono eseguiti il codice virale torna in esecuzione e si diffonde sempre di più [13].

I virus non possono essere inseriti all'interno di file di dati, però va notato che al giorno d'oggi molti programmi possono salvare istruzioni "macro" nei propri documenti, che in alcuni casi possono essere utilizzate per creare gli stessi virus;

- *worm*, diversamente dai virus, non sono pezzi di codice che si agganciano ad altri programmi, bensì programmi che clonano se stessi, lanciandosi in esecuzione in vari modi.

  Spesso i worm contengono del codice "backdoor", il quale installa sulle macchine infettate un accesso per il proprio creatore.

Un'annotazione finale riguarda l'utilizzo da parte degli aggressori di metodologie di attacco non strettamente informatiche: per esempio l'utilizzo di metodologie di social

ISSN: 2279-8773

engineering [14], per ottenere accessi ingannando le persone, oppure tramite il cosiddetto dumpster diving, ovvero la ricerca di preziosi frammenti di informazione, come password e documenti, nella spazzatura eliminata da un'organizzazione.

In questi ultimi due casi, è del tutto possibile che le diverse fasi di attacco illustrate in precedenza vengano completamente by-passate, giungendo direttamente a dei risultati, a volte anche insperati.

Grazie a quest'ultima osservazione, illustriamo dunque in fig.4 lo schema completo dell'incidente informatico come processo.

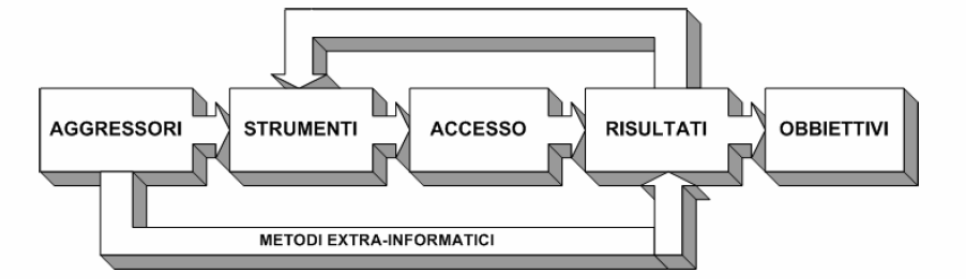

Fig. 4 - Lo schema completo dell'incidente informatico.

ISSN: 2279-8773

Capitolo 2 -

Intrusion Detection e Prevention System

Il concetto di 'Intrusion Detection System' è stato sviluppato da James P. Anderson in un suo rapporto tecnico del 15 aprile 1980 [15].

L'idea alla base di un Intrusion Detection o Prevention System, in letteratura informatica conosciuto come IDS o IPS, è che sia possibile rilevare i segnali caratteristici di un'intrusione in un sistema informatico allo scopo di dare l'allarme.

Quando si progetta la sicurezza di un fabbricato, si comincia a realizzare un sistema di chiusure, ad esempio attraverso porte, lucchetti e chiavistelli, che consenta di "selezionare" le persone che hanno accesso alle varie aree della costruzione.

Dopodiché si integra questo sistema di "difesa" con un

ISSN: 2279-8773

sistema di rilevazione, ovvero di "antifurto", che consenta di individuare eventuali trasgressori.

Un Intrusion Detection System (IDS) è l'equivalente informatico di un antifurto; un Intrusion Prevention System (IPS) invece, aggiunge alle già citate funzionalità di un IDS la possibilità di prevenire tali accessi non autorizzati, reagendo dinamicamente all'inizio di un potenziale attacco informatico.

Dopo aver introdotto il concetto con queste analogie, vediamo però di esplorare più approfonditamente cosa siano e come funzionino questi dispositivi.

Abbiamo detto che, a fronte delle possibilità non secondarie di violazione di una policy di sicurezza da noi prestabilita, dovremmo necessariamente trovare dei meccanismi complementari di difesa.

Molto naturalmente questi meccanismi sono quelli di individuazione degli intrusi, reazione, contenimento del danno e "chiusura" del buco nel sistema difensivo che ha reso possibile la compromissione.

Ovviamente gli IDS e gli IPS entrano in gioco proprio nella

ISSN: 2279-8773

difficile fase di "individuazione" degli intrusi.

Se la protezione dalle intrusioni è difficile perché deve essere tenuto conto della determinazione da parte dell'intruso a violare le politiche di sicurezza, a maggior ragione l'individuazione di queste violazioni sarà doppiamente difficile.

A meno di clamorosi autogol infatti, uno dei primissimi obiettivi di qualsiasi attaccante è quello di coprire le proprie tracce, cercando di comportarsi nella maniera più anonima possibile e soprattutto distruggendo qualsiasi prova della violazione perpetrata.

Il problema maggiore è che, laddove la sicurezza del sistema sia stata completamente sovvertita e l'intruso abbia raggiunto i privilegi amministrativi, improvvisamente tutti i sistemi di registrazione degli eventi e tutti i diversi controlli inseriti nel sistema possono ritenersi terribilmente inaffidabili in quanto suscettibili di ogni possibile manipolazione.

Proprio per questo motivo gli amministratori più accorti hanno l'abitudine di collegare il logging dei sistemi critici a

ISSN: 2279-8773

una stampante.

In questo modo, l'intruso può al massimo disabilitare il sistema di logging, ma non cancellare selettivamente o modificare le proprie tracce.

Sono però evidenti i limiti di questo tipo di artifici difensivi; per questo motivo è necessario che il sistema che tenti di individuare queste violazioni adotti un'ottica il più possibile diversa da quella utilizzata per l'implementazione del sistema di sicurezza.

Laddove quest'ultimo tradizionalmente si basa, come abbiamo detto, su identificazione e privilegi ("Chi sei? Cosa sei autorizzato a fare?"), un IDS o un IPS cercano piuttosto di capire "che cosa stai cercando di fare?", o "perché stai operando in questo modo?".

In altre parole un sistema informatico ha uno scopo determinato a priori, per cui è stato progettato.

Il paradigma CID è un metodo per controllare il sistema in maniera tale che questo scopo possa essere compiuto in modo affidabile, mediante la costruzione della relazione tra risorse e utenti.

ISSN: 2279-8773

Ma la finalità del sistema informatico è in realtà proprio lo scopo originario, e il paradigma CID e la sua specifica sono soltanto dei mezzi, non delle finalità in se stesse.

Quindi ciò che idealmente un IDS o un IPS dovrebbero fare è "individuare quelle azioni che cercano di fare agire il sistema in un modo diverso rispetto a quello per cui è stato progettato".

Ciò significa, principalmente, cercare di cogliere quelle azioni tese ad eludere le misure di sicurezza, nella speranza che allo stesso tempo esse siano sensibilmente differenti rispetto a quelle degli utenti autorizzati.

È evidente che un paradigma di questo tipo rappresenta un approccio complementare alla sicurezza rispetto a quello proposto dal paradigma CID.

Tornando all'esempio dell'antifurto, se il paradigma CID equivale a lucchetti e porte, mentre invece l'IDS o l'IPS sono l'equivalente dell'antifurto, nessuno installerebbe mai tale antifurto senza mettere anche delle buone serrature.

Progettare dispositivi di questo genere dovrebbe essere un passo "finale" nel lungo percorso per rendere finalmente

ISSN: 2279-8773

sicuro un sistema informativo, che non esime affatto alcun amministratore dalla necessità di un'ottima sicurezza di tipo tradizionale.

In effetti, i due paradigmi sussistono per essere quanto più possibile indipendenti, nella speranza di rendere più difficile la compromissione di entrambi.

Come nota a margine, è evidente che il concetto di Intrusion Detection e Prevention System da noi tracciato in questo paragrafo è paradigmatico.

Non si sta parlando di un singolo software, prodotto o meccanismo, ma dell'intera architettura di sistemi che vengono predisposti per rilevare le intrusioni.

2.1 I concetti principali e lo stato dell'arte

Esistono in letteratura due grandi categorie di approcci alla tematica dell'Intrusion Detection e Prevention:

- *anomaly detection*, in questo tipo di approccio si cerca di studiare il comportamento dell'utente, confrontandolo con un profilo di comportamento "normale", modellato secondo varie tecniche.

ISSN: 2279-8773

Il sistema individua in modo statistico qualsiasi deviazione significativa, segnalandola all'amministratore come "sospetta".

I sistemi di questo tipo vengono anche detti "behavior based";

- *misuse detection*, in questo tipo di approccio, viceversa, il sistema cerca di individuare direttamente un comportamento "anomalo", solitamente basandosi su qualche forma di base di conoscenza che contenga un catalogo di attacchi noti, come ad esempio "firme" o "signatures"; i sistemi di questo tipo vengono anche detti "knowledge based".

Ciascuno di questi approcci ha pregi e difetti: i sistemi basati sulla anomaly detection, per esempio, non richiedono un'immissione di conoscenza "a priori", né richiedono continui aggiornamenti delle "firme" d'attacco, essendo teoricamente in grado di rilevare i "cattivi comportamenti" sulla base di una descrizione di "normalità".

Tuttavia, per costruire un modello di "comportamento

ISSN: 2279-8773

normale" serve innanzitutto uno studio architetturale preciso su quale tipo di modello usare, ed in secondo luogo una fase più o meno prolungata di addestramento in cui il modello viene "tarato" sullo specifico utente piuttosto che sul particolare sistema.

Inoltre, questi sistemi sono drammaticamente sensibili ad errori e falsi positivi, per i quali daremo delle definizioni più chiare ed esaustive nel seguito.

I sistemi basati sulla misuse detection, viceversa, richiedono uno studio estensivo delle forme d'attacco per la produzione delle "firme" necessarie al loro funzionamento.

Dalla qualità delle firme e dal loro aggiornamento costante dipende inevitabilmente l'efficacia del sistema: si può quindi notare immediatamente un parallelo con i meccanismi e le problematiche tipiche dei software di antivirus.

Stranamente, il problema dei falsi positivi, ma soprattutto degli alert indesiderati, affligge drammaticamente anche i sistemi di questo tipo, che invece dovrebbero essere meno vulnerabili.

ISSN: 2279-8773

Ciò che si nota infine, è che mantenere una base di conoscenze vasta e in continuo aggiornamento con i "pattern" degli attacchi è un'impresa improba e spesso inutile.

Alcuni sistemi commerciali ed accademici di Intrusion Detection e Prevention combinano entrambi questi aspetti, in quanto ciascuno di essi possiede delle caratteristiche molto desiderabili, e sono a tutti gli effetti complementari.

Vi è tuttavia un problema di metriche decisionali, e di aumento incontrollato dei falsi positivi, che ha scoraggiato fino ad ora lo sviluppo di simili soluzioni.

Nella stragrande maggioranza dei casi, i sistemi di Intrusion Detection e Prevention commerciali sono "misuse based".

Un'altra classificazione molto importante distingue i sistemi di anti-intrusione tra quelli host-based e quelli network-based:

- *i sistemi "host-based"* controllano una singola macchina ed a volte una singola applicazione, e dipendono dal sistema operativo a cui sono spesso collegati

ISSN: 2279-8773

strettamente (nello specifico molti moduli di Intrusion Detection e Prevention host based sul sistema linux vengono realizzati come moduli del kernel), per tracciare chiamate di sistema, utilizzo delle risorse, comandi eseguiti e i passaggi da un livello di privilegi a un altro.

- *i sistemi "network-based"* viceversa, sono collegati in posizioni opportune ad una rete di computer, e tentano di controllare tutto il traffico che la attraversa, ad esempio mediante degli sniffer di rete, cercando nel flusso di pacchetti le indicazioni di possibili attacchi.

Come sempre, entrambi gli approcci presentano vantaggi e svantaggi.

Uno dei vantaggi principali di un IDS/IPS network-based (da ora in poi solamente IPS) è la possibilità di usare un numero di sonde relativamente piccolo per controllare anche reti di grandi dimensioni.

Vi sono però alcuni tipi di intrusione che un sistema network-based non può, realisticamente, individuare: esempi tipici sono tutte le attività che coinvolgono canali

ISSN: 2279-8773

crittografici.

In pratica un IPS di rete può certo intercettare ed analizzare i tentativi di attacco rivolti ad una web application, per esempio individuando i dati mal formattati inviati ad un campo di una form, ma se questi ultimi viaggiassero attraverso una sessione criptata, per esempio mediante il protocollo Secure Socket Layer (SSL), le "firme" dell'IPS di rete si rivelerebbero sostanzialmente inutili.

Uno dei potenziali vantaggi di un IPS network-based è la possibilità di operare una "passive analysis", in cui il sistema raccoglie i pacchetti di rete senza che la sua presenza sia rilevabile: operativamente questo si ottiene dotando la macchina che esegue l'IPS di una scheda di rete posta in modalità promiscua e con un indirizzo IP nullo, ovvero `0.0.0.0`, in maniera tale che essa non possa trasmettere alcunché, ma che invece riceva ed analizzi qualsiasi pacchetto in transito indirizzato ad un qualsiasi altro indirizzo.

In questo modo diventa molto difficile per un intruso rilevare la presenza dello sniffer.

ISSN: 2279-8773

Tuttavia una serie di ricerche [16] hanno dimostrato la possibilità di rilevare comunque la presenza di interfacce poste in modalità promiscua.

Fino a dieci anni fa i sistemi più importanti erano quelli host-based; tuttavia, l'avvento dell'era di Internet ha reso la rete uno dei vettori d'attacco più facili e probabili, dando notevole impulso allo sviluppo di sistemi network-based, il primo dei quali, storicamente, è stato Network Security Monitor, o NSM [17].

Anche questi due tipi di approcci, in realtà, non sono contrapposti ma altamente complementari.

Tutte le suite per IDS più popolari al mondo, tra cui RealSecure di ISS o i sistemi IDS di Cisco Systems, combinano sonde di tipo host-based e sonde di tipo network-based per cercare di coprire quanti più tipi d'attacco possibile.

Sicuramente nel futuro sarà sempre più importante un approccio misto di questo tipo, che presenta interessanti sfide per la normalizzazione dei dati derivanti da sensori diversi, per la correlazione tra essi, e per lo sviluppo di

ISSN: 2279-8773

sistemi in grado di ragionare su queste viste correlate per fornire analisi di intrusione di più alto livello.

Si parla quindi di applicare al campo della Intrusion Detection e Prevention i concetti di "multisensor data fusion" già in corso di studio nel campo, per esempio, della robotica [18].

Cerchiamo ora di spiegare la distinzione fra sistemi on-line, distinti tra sistemi in-line e real-time, e sistemi off-line.

Su questi termini ci soffermiamo perchè essi rappresentano spesso fonte di confusione.

Vengono definiti "on-line" gli IPS che analizzano i dati durante lo svolgimento delle azioni, mentre i sistemi di tipo "off-line" o batch analizzano i dati a posteriori.

Ultimamente si tende a considerare questi ultimi più come dei tool di "computer forensics", ovvero di analisi a posteriori delle intrusioni con scopi generalmente di polizia giudiziaria, piuttosto che programmi di "intrusion detection" in senso stretto.

In ogni modo, nella definizione che si è data, nel presente lavoro di tesi, di "Intrusion Detection e Prevention System"

ISSN: 2279-8773

come un insieme di accorgimenti mirati a rilevare, identificare e contenere le intrusioni, è indubbio che sistemi di questo tipo siano da considerarvisi totalmente inclusi.

Inoltre all'interno dei sistemi on-line spesso si usa la definizione "real-time" per indicare che l'IPS è progettato per gestire un determinato flusso di dati senza perdere pacchetti e senza rimanere con un "buffer" in attesa di essere verificato.

I sistemi "in-line" invece sono i sistemi di Intrusion Detection e Prevention posizionati come un firewall o uno switch sulla "traiettoria" dei pacchetti, e la sua funzione più ovvia è quella di poter fungere anche da filtro oltre che da analizzatore.

Un IPS può quindi essere costituito da più processi eseguiti su varie macchine, denominati sensori.

Un esempio notevole in campo accademico, forse uno dei primi e più documentati, è DIDS, acronimo di Distributed Intrusion Detection System [19].

A volte, quando tali macchine sono dedicate esclusivamente al software di Intrusion Detection o

ISSN: 2279-8773

Prevention, vengono semplicemente chiamate sensori o sonde, e vengono spesso realizzate sotto forma di appliance integrate composte da hardware e software proprietario, ottimizzato e preinstallato.

Un IPS così strutturato può quindi utilizzare un modello distribuito o centralizzato per l'analisi dei dati e per la loro collezione a seconda che i vari sensori procedano per conto proprio o facciano riferimento ad un server centrale.

Di fatto le due cose possono essere ben miscelate: infatti le sonde potrebbero analizzare ciascuna i propri dati per poi trasmettere i risultati ad un server centrale per tutte le successive rielaborazioni.

I sistemi di IPS distribuiti sono stati efficacemente descritti e progettati come sistemi basati su agenti, ad esempio nell'eccellente progetto 'Autonomous Agents for Intrusion Detection' (AAFID) [20], sviluppato presso il gruppo COAST dell'università Purdue.

2.2 Le caratteristiche desiderabili per un IPS

Una delle caratteristiche più apprezzate di un IPS è quella

ISSN: 2279-8773

di poter in qualche modo reagire agli attacchi, ovvero cercare di bloccarli automaticamente.

Di fatto un sistema off-line non ha nessuna reale possibilità di reagire ad un'intrusione prima che si verifichino dei danni.

Se poi anche il sistema fosse on-line, l'unico modo efficace per consentirgli di operare preventivamente sarebbe quello di porlo in posizione "in-line", come filtro pregresso a tutto il traffico.

Ciò risulta fattibile, ma tuttavia considerazioni di performance possono rendere questa soluzione irragionevole su linee ad alta capacità.

Una soluzione alternativa è un IDS on-line, real-time di tipo tradizionale, in qualche modo accoppiato con un firewall o con le ACL dei router di frontiera, in grado quindi di interrompere la comunicazione tra l'host vittima e l'attaccante, come esposto in questo lavoro di tesi.

Un'altra soluzione che esula dal campo che si sta studiando, ma dalle interessanti applicazioni, è Angel, un Loadable Kernel Module (LKM), sviluppato da due studenti

ISSN: 2279-8773

italiani, che cerca di impedire agli utenti di una macchina di utilizzarla per lanciare attacchi; una sorta di Intrusion Prevention, ma in uscita.

In campo militare sono stati velatamente proposti anche IDS reattivi con potenzialità di contrattacco verso l'host aggressore, ma l'idea è stata saggiamente scartata, almeno ufficialmente, per le ovvie e incontrollabili implicazioni.

Un IPS può essere poi progettato per la "survivability": dal momento che, esattamente come per la scatola nera di un aereo, le sue registrazioni possono essere l'unico mezzo per stabilire cosa sia accaduto su una rete, diventa importante garantire che il sistema stesso sia meno vulnerabile possibile.

La sovversione di un sistema di difesa è infatti il più pericoloso risultato di un attacco telematico, in quanto lascia agli attaccati un falso senso di sicurezza.

Deve inoltre essere in qualche modo garantita la sicurezza della comunicazione tra le varie componenti dell'IPS; pertanto opportuni meccanismi di autenticazione, crittografia, e verifica di integrità devono essere

ISSN: 2279-8773

implementati per evitare ogni sorte di attacchi di spoofing o di tipo man-in-the-middle.

È inoltre opportuno che, per ogni tipo di attacco ed aggressione, come ad esempio la distruzione di una sonda o flood di pacchetti per disturbare l'analizzatore di rete, il sistema presenti robustezza, ovvero non si arresti completamente ma esibisca, eventualmente, una graduale riduzione delle performance man mano che l'attacco procede.

All'interno di un Network Operations Center (NOC) o di un Security Operations Center (SOC) spesso vi sono presenti pochi tecnici, specie durante i turni notturni, magari non eccessivamente esperti, che devono sovrintendere al controllo di numerose reti remote che conoscono solo a grandi linee.

Oltre alle ovvie problematiche connesse ai falsi positivi, i sistemi di IPS dovrebbero essere progettati per comunicare quante più informazioni possibili al personale che deve controllarli, nel modo più chiaro ed intuitivo, possibilmente fornendo anche una guida alla soluzione di un determinato

ISSN: 2279-8773

problema.

In aggiunta, deve essere possibile una totale flessibilità ed adattabilità del sistema a circostanze molto specifiche di ogni singola rete.

Questo può essere garantito in modo automatico da un IPS in modalità anomaly detection, ma deve anche essere opportunamente studiato per sistemi basati su firme.

Ad oggi, i possibili tipi di attacco contro una rete informatica evolvono in continuazione.

Si può quindi distinguere una microevoluzione, ovvero un processo con cui vengono ideati nuovi attacchi all'interno di categorie di attacchi già esistenti, ed una macroevoluzione, mediante la quale vengono scoperti, più lentamente, dei metodi di attacco completamente nuovi.

È abbastanza evidente che la macro evoluzione può creare forme d'attacco per cui il sensore non è architetturalmente predisposto.

Per fare un esempio, se il sensore controlla il traffico di rete al layer 3 ISO/OSI, e venisse sviluppata una metodologia di attacco che coinvolge manipolazioni al layer 2, è molto

ISSN: 2279-8773

probabile che un semplice aggiornamento della base di conoscenza o del modello con cui l'IPS opera non sia sufficiente.

In questo caso, si renderebbe necessaria, con ogni probabilità, una revisione del motore di analisi in sé.

Viceversa, in linea di principio, se una forma di attacco è già nota, le sue successive microevoluzioni dovrebbero risultare relativamente semplici da inserire nella base di conoscenze o nel modello dell'IPS.

Alla luce di ciò, potrebbe sembrare che il problema di aggiornamento colpisca soprattutto i motori di misuse detection, che in effetti operano su una base di firme che tende a diventare sempre meno adeguata con il tempo, ma è tuttavia importante sottolineare che anche i sistemi di anomaly detection di fatto condividono questo problema.

Infatti la scarsa diffusione questi sistemi ha reso fino ad ora poco studiate le possibili tecniche di evasione, ma è certo possibile, per alcuni versi persino facile, immaginare delle forme di attacco che sfruttino gli "angoli morti" di queste soluzioni, ovvero tecniche di aggressione i cui gli unici

ISSN: 2279-8773

sintomi ricadano su quelle variabili che sono state scartate come indicatori di normalità in fase di progettazione.

È abbastanza evidente come in questo caso l'unica soluzione sarebbe quella di ristudiare il modello da capo.

Inoltre, un buon sistema IPS deve essere in grado di scalare seguendo la crescita della rete.

Per questo motivo, le soluzioni network-based hanno spesso sofferto rispetto a soluzioni host-based, in quanto analizzare gli eventi in modo distribuito è sicuramente più semplice che cercare di capire qualcosa analizzando tutto il traffico della rete aggregato.

Tuttavia va notato che comunque, a livello commerciale, esistono prodotti ed appliance in grado di supportare volumi di traffico dell'ordine dei Gigabit e che inoltre esistono studi accademici relativi alle problematiche di implementazione di sistemi di misuse detection ad altissime prestazioni, quale ad esempio BRO [21].

Tipicamente, quando un IPS di rete supera la propria capacità di analizzare i pacchetti, inizia a lavorare in una modalità "a campione", scartando parte del buffer con una

ISSN: 2279-8773

tecnica simile a quella utilizzata dai router.

Il problema è che, nel caso dei router, i meccanismi di re-trasmissione del TCP evitano la perdita di pacchetti, mentre nel caso dell'IPS, a meno di fortunate coincidenze, ciò che si è perso si è perso, e malauguratamente potrebbe trattarsi proprio dei pacchetti che interessano l'attacco.

Se tale coincidenza pare troppo azzardata e teorica, si provi a guardare la cosa dal punto di vista dell'attaccante: generare una marea di traffico di nessun interesse è relativamente semplice; se ciò significasse mettere in crisi l'IPS ancora prima che l'attacco sia cominciato, si può essere sicuri che qualsiasi attaccante con un minimo di esperienza adotterà questa tecnica, nella speranza che ad essere scartati siano proprio i pacchetti con le firme d'attacco.

Questo tipo di attacco viene spesso citato dai venditori di strumenti in-line come uno dei motivi per preferire la loro soluzione.

A volte questo può risultare vero, ma non va trascurata la possibilità che questa "feature" si trasformi in un comodo

ISSN: 2279-8773

mezzo per perpetrare un attacco di tipo denial-of-service.

In aggiunta ai problemi di scalabilità, il sistema deve essere in grado di gestire attacchi multipli concorrenti.

Molto raramente infatti un intruso proverà un singolo attacco contro le nostre macchine: spesso utilizzerà uno strumento che tenta attacchi multipli, e quindi si vuole essere in grado di seguirli pressoché tutti.

Non solo, sulla scia delle stesse considerazioni fatte in precedenza, se l'attaccante sapesse che il nostro IPS fosse in grado di seguire soltanto un attacco alla volta potrebbe far in modo di generare un finto attacco proveniente da chissà dove per "distrarre" il nostro sistema di rilevazione.

2.3 I problemi tipici di un IPS

Fino ad ora abbiamo analizzato l'adattabilità di un IPS a forme di attacco nuove, presupponendo l'intervento umano nell'aggiornamento di un'eventuale base di conoscenza di un sensore.

Uno dei problemi fondamentali che abbiamo volutamente trascurato è la possibilità, del tutto realistica, che nuove

ISSN: 2279-8773

forme di attacco colpiscano i sistemi prima che gli esperti abbiano occasione di analizzarle.

Nella tassonomia degli attaccanti abbiamo infatti distinto coloro che si limitano ad utilizzare strumenti di attacco ben conosciuti e diffusi, come gli script kiddies, da coloro che viceversa sono in grado di creare nuovi attacchi.

Ovviamente, nel primo caso è prevedibile che sia disponibile una "firma" per quel particolare tool d'attacco, mentre nel secondo caso non è così certo.

Pertanto, dobbiamo valutare se l'IPS sia o meno in grado di far fronte autonomamente all'evoluzione degli attacchi, o se dipenda totalmente dall'aggiornamento manuale della sua base di conoscenze.

Come premessa, possiamo presupporre che le nuove forme di attacco sottoposte ad un IPS facciano parte della microevoluzione.

Nuove macro categorie di attacchi sono rare e generalmente vengono scoperte e portate alla luce molto prima che strumenti di attacco effettivi ed efficaci vengano sviluppati nell'underground informatico.

ISSN: 2279-8773

Ciò premesso, è abbastanza evidente che un IPS che dipende da una base di conoscenze sia molto meno resistente, almeno da un punto di vista teorico, alla microevoluzione degli attacchi rispetto ad un sistema basato sull'individuazione di anomalie.

Esiste poi un problema sostanzialmente inaffrontabile, che è quello del polimorfismo degli attacchi.

Infatti in molti casi è possibile raggiungere un determinato scopo seguendo una pluralità di metodi, ed è corrispondentemente più difficile sviluppare delle firme appropriate.

Per fare due esempi classici, tutti i tipi di bug connessi all'uso di caratteri "unicode" sono inerentemente polimorfi, in quanto esistono codifiche multiple per ogni singolo carattere: ciò significa una crescita combinatoria nel numero di firme necessarie ad intercettare tutte le possibili variazioni di questi attacchi.

Un secondo esempio, ben più inquietante, è dato dallo strumento ADMutate [22], sviluppato da "K2", un hacker canadese il cui nome reale è ignoto: si tratta di uno

ISSN: 2279-8773

strumento in grado di criptare lo shellcode di un attacco di buffer overflow destinato allo stack-smashing.

Come si è già accennato, questo tipo di bug consiste nella possibilità per l'aggressore di scrivere in un buffer limitato una stringa di lunghezza arbitraria, andando a sovrascrivere lo stack del programma, quindi in pratica assumendo il controllo del flusso di esecuzione.

Una stringa di buffer overflow, in buona sostanza, consiste in una sequenza lunghissima di caratteri a cui segue una serie di istruzioni in codice macchina, il cosiddetto "shellcode".

L'idea è che la stringa riempia il buffer, debordi, e che si arrivi a sovrascrivere il puntatore alla successiva istruzione presente sullo stack, facendolo puntare proprio allo shellcode e consentendo così l'esecuzione di comandi arbitrari.

Siccome molto spesso lo shellcode contiene delle porzioni di codice "comuni", vi sono firme "generiche" sugli IPS misuse based che si propongono di intercettarle.

ADMutate cripta lo shellcode, anteponendo ad esso la

ISSN: 2279-8773

routine di decriptazione; in tal modo sarà possibile ad ogni esecuzione dell'attacco variare la forma, ma non l'efficacia, del codice macchina utilizzato, mandando di fatto in confusione un discreto numero di sistemi IPS network-based commerciali.

Anche se ad oggi praticamente tutti gli IPS contengono un'apposita firma per riconoscere ADMutate, ciò non toglie che lo stesso principio possa essere in linea teorica ripetuto all'infinito.

Il polimorfismo è di fatto un avversario strutturale di qualsiasi sistema IPS basato sulla misuse detection, in quanto mina alla base l'assunto che sia possibile elencare le forme d'attacco.

Molto spesso la risposta dei sostenitori della misuse detection a questa obiezione è che si possono trovare firme più intelligenti per gli attacchi.

Si tratta però di un eterno gioco di scacchi in cui ad ogni mossa corrisponde una contromossa uguale e contraria.

Gli scacchisti dovrebbero poi sapere che nei "mirror game" in cui ad ogni mossa si risponde con la mossa speculare

ISSN: 2279-8773

sono destinati alla vittoria inevitabile di chi ha mosso per primo.

Fuor di metafora, è inutile continuare in questa direzione: per dare un altro esempio concreto, utilizzando di nuovo gli attacchi di tipo "buffer overflow", si potrebbe osservare che molti IPS prevedono anche una firma in grado di riconoscere lunghe sequenze di `0x90`, che corrisponde al codice esadecimale dell'operazione `NOP` su processori Intel X86.

Questo codice viene tipicamente usato nei buffer overflow come "riempimento" in quanto è spesso difficile determinare il punto esatto dal quale tale codice verrà eseguito.

Per evitare di essere riconosciuto in questo modo, un attaccante più avanzato potrebbe, per esempio, utilizzare un'istruzione di `JUMP` all'indirizzo successivo (`0xeb 0x00`) al posto della `NOP`, con il solo problema aggiunto di utilizzare un codice a 2 byte (sarebbe possibile che l'esecuzione inizi "a metà" di un'istruzione quindi fallendo, ma in tal caso si potrebbe rimediare facilmente spostando di uno l'offset del

ISSN: 2279-8773

codice).

Se anche questa firma venisse aggiunta all'IPS, si può iniziare a giocare di nuovo con i salti, in un'eterna rincorsa che i "buoni" sono destinati inevitabilmente a perdere.

Gli errori di un IPS vengono quindi tipicamente suddivisi in due categorie.

Si parla di "falso positivo" quando l'IPS segnala come anomala un'azione che è legittima o innocua.

Si parla di "falso negativo" quando viceversa un IPS non suona l'allarme in presenza di un'azione evidentemente maliziosa.

I "falsi positivi" possono suonare innocui: in fondo, meglio essere avvertiti una volta di troppo che una volta di meno.

La famosa favola del ragazzino che gridava "al lupo!" ci ricorda che questo può non essere sempre vero: difatti un IPS che segnali in continuazione come "pericolose" azioni perfettamente normali dopo un pò verrà ignorato.

Eliminare i falsi positivi è dunque uno dei focus della ricerca sugli IPS, in particolare per i sistemi di anomaly detection in quanto a meno di errori nelle firme è molto difficile che

ISSN: 2279-8773

un sistema di misuse detection fornisca dei falsi positivi.

Va infatti notato che spesso quelli che gli utenti finali, in particolare nei sistemi di misuse detection, indicano come "falsi positivi" sono in realtà le indicazioni di attacchi, veri e reali, contro piattaforme software o hardware che essi non hanno.

Ciò che l'IPS sta segnalando, in quel caso, non è un falso positivo, ma è piuttosto un vero attacco che qualcuno, alla cieca, ha rivolto contro le macchine sbagliate.

In tal caso, sta all'interesse dell'amministratore scegliere se essere avvertito di tali attacchi, che per quanto innocui, non sono comunque un buon sintomo, o ignorarli direttamente durante la fase di tuning del sistema.

I falsi negativi sono invece un dramma aperto: si tratta del mancato riconoscimento di una forma d'attacco.

In generale, in un sistema basato sulla misuse detection, un falso negativo è associato alla mancanza di una firma per l'attacco effettuato, ricadendo quindi sul problema dell'individuazione di nuovi attacchi precedentemente esposto.

ISSN: 2279-8773

In un sistema basato sull'individuazione di anomalie invece, è più probabile che si abbia un falso negativo magari anche su un attacco che in precedenza era stato riconosciuto, proprio a causa della natura statistica di questo tipo di sistemi.

In aggiunta, il fattore tempo presenta una delle più grandi incognite per un progettista di IPS.

È abbastanza evidente, ad esempio, il problema di ricostruire la sequenza tra eventi di qualsiasi tipo che siano stati rilevati da sensori dispersi su una rete di notevoli dimensioni.

Infatti, il tempo di propagazione dei messaggi e la natura "best effort" delle reti TCP rende impossibile basarsi esclusivamente sull'ordine di arrivo di questi pacchetti ad un server centrale, sempre ammesso che un server centrale esista.

Sono ben noti i problemi che si incontrano nel cercare di dotare di un tempo unico un gran numero di macchine in rete, magari anche utilizzando protocolli ben studiati come Network Time Protocol (NTP).

ISSN: 2279-8773

Tuttavia questa è solo la punta dell'iceberg dei problemi che il fattore tempo impone ad un progettista di IPS.

Innanzitutto infatti, non è detto che un attacco sia costituito da un'azione atomica: in realtà moltissimi attacchi sono costituiti da più eventi indipendenti, ciascuno dei quali preso singolarmente può risultare innocente o quantomeno innocuo.

Non è nemmeno detto che questi eventi debbano avvenire entro un determinato intorno di tempo l'uno dall'altro, o che non possano essere inframmezzati da un numero arbitrario di altri eventi assolutamente indipendenti (problema dell'interleaving).

Non solo, è inoltre ampiamente possibile che a tutto questo si aggiungano problemi di unificazione, ovvero la presenza all'interno di una sequenza di attacco di un elemento variabile che può assumere un'infinità di valori.

Per finire, non è assolutamente detto che tali sequenze siano strettamente ordinate; molto spesso infatti si tratta di un ordinamento assolutamente parziale.

Tutto ciò avviene usualmente riassunto nel concetto di

ISSN: 2279-8773

"uncertain reasoning", ovvero l'incapacità di determinare algebricamente se è possibile "chiudere" l'analisi o se è necessario rimanere in attesa del completamento di un attacco.

Per esemplificare, se sappiamo che un attacco è formato dalle fasi A, seguita da B e da C in un ordine qualsiasi, e conclusa da D, con qualsiasi intermissione di eventi, l'osservazione della sequenza A-x-C-B-x-x-x non ci dice nulla sul fatto che si sta osservando un attacco parziale, dal quale dovremmo aspettarci la conclusione, oppure siamo in presenza di una sequenza assolutamente innocente di comandi, sempre nell'ipotesi che A, B e C non siano di per sé delle violazioni.

Il sistema quindi dovrebbe lasciare questa sequenza candidata "appesa", attendendo che avvenga un evento che dimostri che essa non potrà mai soddisfare la regola, o viceversa aspettando che essa necessariamente si completi.

In un sistema realistico questo significa che prima o poi dovrà esserci un timeout o un qualche altro meccanismo

ISSN: 2279-8773

che consideri scadute le osservazioni molto vecchie per fare spazio a quelle nuove, altresì si offrirebbe ad un attaccante un metodo per disabilitare l'IPS intasando la memoria con sequenze appese.

Ovviamente tutti questi problemi sono tipici dei sistemi on-line, che soffrono contemporaneamente di "incertezza del futuro" e di una finestra limitata del passato, per ovvie ragioni di performance.

In questo caso sono enormemente avvantaggiati i sistemi off-line, che possono di fatto gestire una finestra illimitata nel passato, e che conoscono già l'arco temporale futuro prefissato.

Un sistema IPS network-based inoltre deve affrontare un ulteriore insieme di problemi derivanti strettamente dalla sua fonte di informazione, ovvero il traffico di rete.

Per discutere di queste problematiche facciamo riferimento alle reti basate su TCP/IP, ma il discorso è molto generale e si potrebbe estendere anche ad altri tipi di rete.

In linea teorica, osservando tutto il traffico di rete dovrebbe essere possibile, per un sistema arbitrariamente potente,

ISSN: 2279-8773

tenendo conto della topologia della rete e di tutte le particolarità delle macchine che la compongono, stabilire esattamente quale sia l'effetto di ogni pacchetto IP che viaggia sulla rete.

Tuttavia questa visione risulta inguaribilmente ottimistica: è infatti computazionalmente proibitivo pensare di ricostruire su una singola macchina gli esatti effetti che il traffico di rete possa produrre su ciascuna delle altre macchine connesse; inevitabilmente, un IPS network-based deve cercare altri tipi di approcci per determinare cosa sia un attacco e cosa invece no.

Per esempio, alcuni IPS si limitano ad eseguire un pattern matching pacchetto per pacchetto: questa è senza dubbio una soluzione banale che perde gran parte della relazione temporale e di connessione tra i dati.

Altri sistemi invece, più evoluti, offrono un'analisi di tipo "stateful", ovvero tengono traccia delle connessioni stabilite, ed analizzano i pacchetti basandosi su queste informazioni.

Tuttavia, questa soluzione non è certo il rimedio a tutti i

ISSN: 2279-8773

mali: infatti come è stato dimostrato da numerosi ricercatori [23] può essere molto difficile, se non impossibile, stabilire esattamente, in un qualsiasi punto della rete, quali dei pacchetti intercettati raggiungeranno poi un altro determinato punto.

Su questo principio si basano quindi gli attacchi speculari di inserzione e di evasione dei pacchetti.

L'attacco di inserzione dei pacchetti si basa su quest'idea: se l'IPS fosse stato istruito per dare l'allarme quando osserva la stringa "ATTACCO", una delle possibili soluzioni potrebbe essere quella di spezzettare questa stringa in modo tale che all'IPS giunga qualcosa che per esempio si legga "ATTrACCO", facendo in modo quindi che il pacchetto contenente la lettera 'r' arrivi soltanto all'IPS stesso ma non al sistema attaccato.

Questo si può ottenere, ovviamente solo in casi particolari, manipolando opportunamente le flag del TCP.

Inoltre è di solito molto difficile che gli IPS di rete scelgano di impiegare del tempo a controllare i checksum dei pacchetti, ed un attaccante potrebbe, in diversi modi,

ISSN: 2279-8773

trarre diversi vantaggi anche da quest'aspetto.

Altre ambiguità sulle quali un attaccante potrebbe ragionare sono il campo "Time To Live" (TTL), che potrebbe essere grande a sufficienza per raggiungere l'IPS ma non il sistema bersaglio, l'utilizzo di un pacchetto decisamente 'grande' con attivata la flag TCP "don't fragment" (DF), piuttosto che l'utilizzo dei campi di opzione o timestamp strani, o malformati, per tentare di sfruttare le ambiguità che esistono nelle implementazioni dei vari stack TCP/IP presenti sui diversi sistemi operativi.

Tornando al nostro esempio, l'esatto contrario darebbe quindi origine ad attacchi di tipo "evasion", dove invece di vedere la stringa "ATTACCO", l'IPS riceverà solamente la stringa "TACCO".

I meccanismi che possono essere sfruttati per ottenere questo tipo di ambiguità sono gli stessi considerati prima, con un curioso e perverso legame: infatti più si cerca di rendere un sensore di rete resistente all'evasione, più di fatto si accetteranno pacchetti "anomali", dando quindi spazio ad attacchi di inserzione, e viceversa.

ISSN: 2279-8773

Infine, esiste un'ulteriore problematica derivante dalla possibilità che gli attacchi vengano distribuiti su più pacchetti mediante l'uso della frammentazione.

Ormai tutti i principali IPS di rete hanno affrontato con successo questo problema, ma la ricostruzione dei pacchetti frammentati si è rivelato un problema più difficile del previsto, oltre a richiedere un dispendio di risorse computazionali e di tempo poco compatibile con le moderne esigenze di performance.

Si sono notate infatti, nuovamente, numerose incongruità nel modo in cui i vari stack TCP/IP dei diversi sistemi operativi affrontano il problema della frammentazione, spesso non perfettamente rispondenti alle specifiche standard.

È quindi di fatto impensabile cercare di indovinare tutte le possibili interpretazioni del flusso dei pacchetti basandosi esclusivamente su informazioni variabili quali l'architettura della rete piuttosto che la tipologia dei sistemi operativi in uso.

ISSN: 2279-8773

Capitolo 3 -

Architettura del Sistema

In questo terzo capitolo verrà discussa l'architettura del sistema realizzato ed i principali strumenti utilizzati ai fini della produzione del progetto di tesi poi illustrato nel quarto ed ultimo capitolo.

Verranno quindi mostrati i principali software di produzione e di gestione, nonché la struttura del Centro di Elaborazione Dati (CED) della Ripartizione Informatica dell'Università Degli Studi di Perugia sul quale poggia l'intera sperimentazione.

Quindi, la parte sperimentale del presente lavoro di tesi ruota di fatto intorno alla personalizzazione ed alla produzione di nuovi programmi e diverse metodologie di interazione soprattutto, ma non solo, con il primo di questi strumenti che verranno citati, ovvero Snort [24], lo standard de-facto per l'Intrusion Detection e Prevention.

ISSN: 2279-8773

3.1 Snort

Snort è un IDS network-based, basato sostanzialmente sulla misuse detection, sviluppato sotto GPL (GNU Public License), distribuito gratuitamente su Internet in formato open source ed aperto ai contributi della comunità, e quindi di fatto l'ideale come riferimento per un lavoro di tipo scientifico.

Inoltre, Snort è uno dei progetti open source più famosi del mondo, ed uno dei pochissimi esempi di IDS non commerciali che godano di un supporto e di una base installata tali da renderlo un "concorrente" di media fascia di molti sistemi commerciali.

Snort è un sistema multipiattaforma che utilizza le librerie di cattura dei pacchetti Libpcap, le librerie standard "de facto" per la cattura di pacchetti in rete.

L'architettura di Snort comprende tre sottosistemi primari:

- un packet sniffer/decoder altamente performante;
- un motore a regole basato sul pattern matching;
- un sottosistema per il logging, la creazione di report e l'invio di alert all'amministratore di rete.

ISSN: 2279-8773

Concentriamoci ora sul componente di analisi: il linguaggio con cui vengono espresse le regole di Snort è molto semplice ed efficace.

Ecco un esempio di regola:

```
log tcp any any -> 192.168.1.0/24 79
```

Questa regola si attiva quando viene rilevato del traffico TCP, da qualsiasi host, da qualsiasi porta sorgente (tcp any any), verso un host della classe C 192.168.1.0/24, sulla porta 79 (si potrebbe comunque anche scegliere un range di porte, ad esempio 1:1024 in riferimento alle porte privilegiate), e richiede a snort di effettuare il logging del pacchetto.

Altre azioni possibili sono 'pass', ovvero 'ignora', ed 'alert', che avvisa l'amministratore usando il metodo da lui prescelto.

Ovviamente, in questo contesto, specificare esclusivamente la provenienza e la destinazione di una connessione, senza entrare nel merito del contenuto, sarà raramente utile.

ISSN: 2279-8773

Quindi, per specificare tale contenuto e la tipologia dei pacchetti per i quali far scattare l'allarme si dovrà inserire una porzione aggiuntiva della regola, come nell'esempio che segue:

```
alert tcp any any -> 192.168.1.0/24 80
(content:"/cgi-bin/phf"; msg:"Probe del PHF!";)
```

Questa regola andrà ad individuare, nel traffico diretto verso la porta 80 dei nostri web server, il probe per la vecchissima vulnerabilità del PHF, ormai corretta da anni, ma che viene usata per tradizione negli esempi.
Se volessimo poi individuare del traffico in formato binario potremmo per esempio scrivere, mediante l'operatore `||', una regola di questo tipo:

```
alert tcp any any -> 192.168.1.0/24 111
(content:"|00 01 86 a5|"; msg:"mountd access";)
```

Nelle regole si possono anche specificare altre opzioni e parole chiave come ad esempio `minfrag`, per la

ISSN: 2279-8773

frammentazione dei pacchetti, `ttl`, `id`, `dsize` per i campi omonimi, piuttosto che le varie flag del TCP.

Snort conserva le sue regole di detection in una lista linkata bidimensionale per poi applicare un algoritmo di pattern matching estremamente efficiente, come mostrato in fig.5.

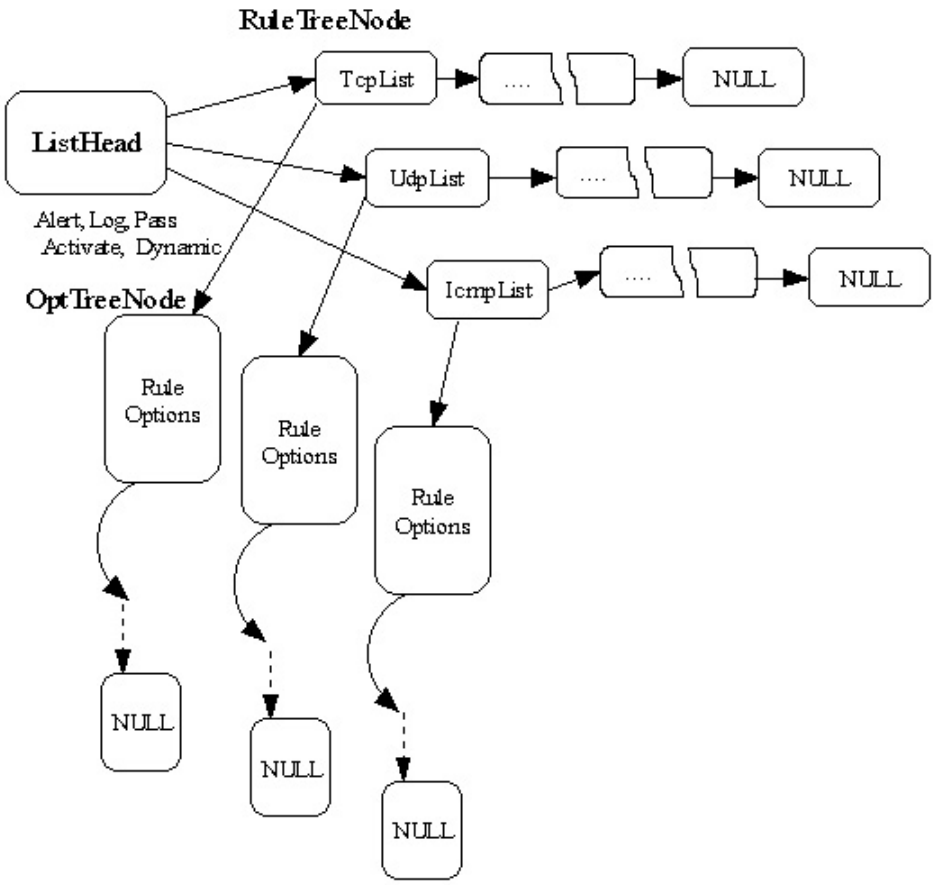

Fig. 5 - Rappresentazione delle regole di Snort.

ISSN: 2279–8773

Riprendendo il concetto di "catena" utilizzato nel packet filtering del kernel di linux, i pacchetti vengono fatti 'combaciare' dapprima in orizzontale sulla base dell'header, ed in seguito in verticale per le opzioni di payload solo in caso di match degli header, in modo da ottimizzare il più possibile il tempo di analisi.

Snort quindi può essere configurato in quattro diverse modalità:

- *sniffer mode*, dove semplicemente vengono letti i pacchetti dalla rete e quindi visualizzati sullo standard output sotto forma di uno stream continuo;
- *packet logger mode*, dove i pacchetti vengono loggati e quindi registrati nel sistema;
- *in-line mode*, dove i pacchetti sono ricevuto da un firewall invece che dalle Libpcap, ovvero dove l'host di Intrusion Prevention è anche il firewall stesso, ad esempio Iptables, posizionato in modalità bridge in modo da consentire o negare i pacchetti in transito;
- *network intrusion detection system (NIDS) mode*, dove il traffico viene realmente analizzato e dove vengono

ISSN: 2279-8773

prese diverse decisioni in base ad un insieme di regole definite dall'utente.

In fig.6 viene mostrato lo schema di funzionamento generale di Snort durante quest'ultima modalità; quindi mediante l'opzione '-D' viene attivata l'opzione che fa partire Snort in modalità daemon.

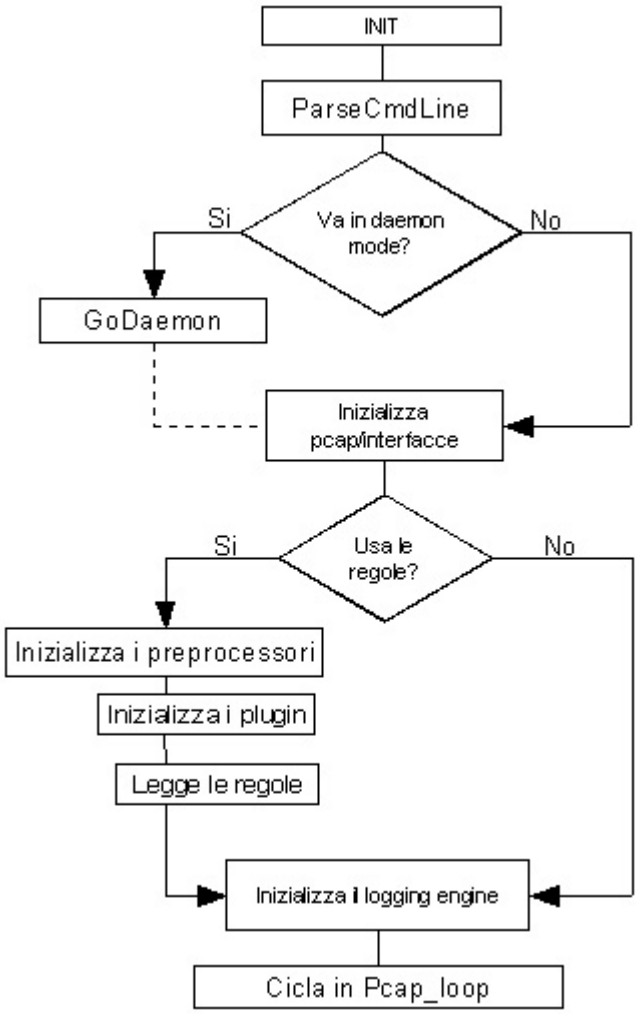

Fig. 6 - Schema a blocchi del funzionamento di Snort.

ISSN: 2279-8773

Dopo che tutte le regole sono state lette e tradotte nel formato interno, viene inizializzato il sistema di logging ed infine viene chiamata la routine `Pcap_loop()`.

La routine `Pcap_loop()` chiama quindi la routine `ProcessPacket()` ogni volta che viene catturato un pacchetto, come mostrato in fig.7.

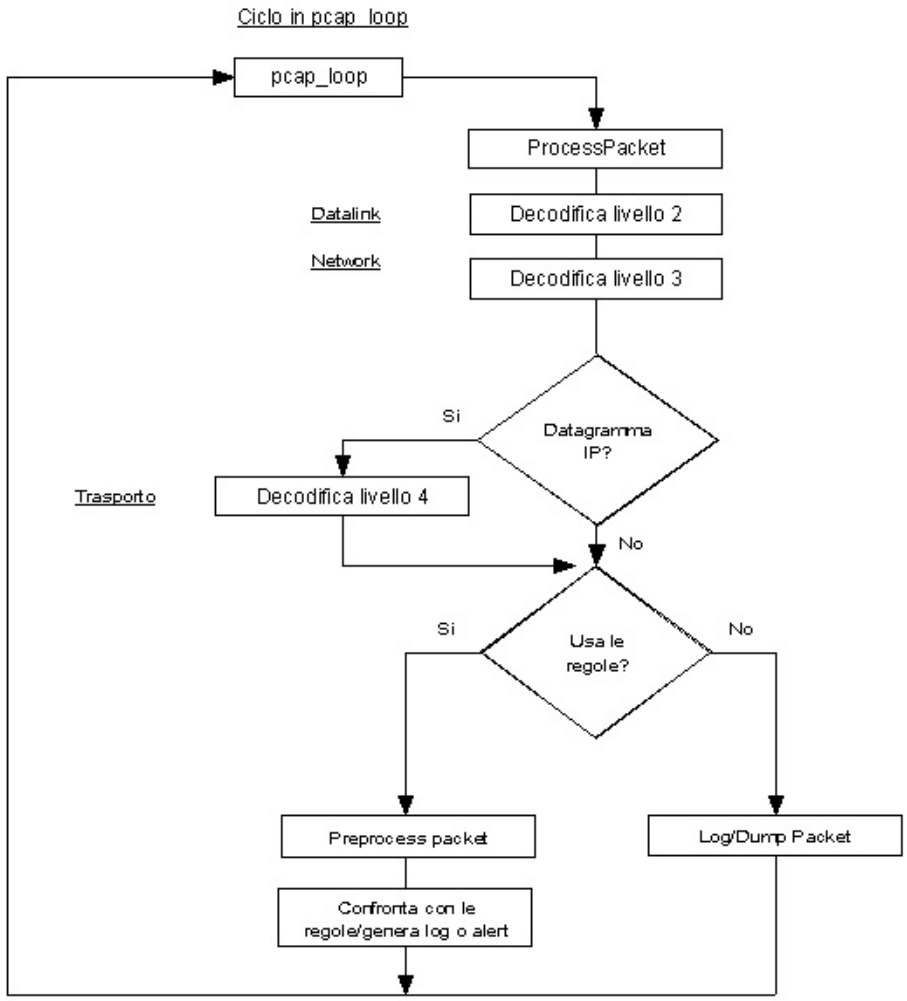

Fig. 7 - Schema a blocchi della routine `Pcap_loop()`.

ISSN: 2279-8773

La routine `ProcessPacket()` è di fatto il punto di ingresso per le diverse routine che svolgeranno un'iniziale decodifica dei pacchetti a seconda dello specifico layer; durante l'intero cammino del pacchetto verranno quindi appropriatamente riempiti i campi della struttura Packet.

Ogni routine, come ad esempio `DecodeEthPacket()`, `DecodePppPacket()` o `DecodeFDDIPacket()`, riempirà i campi della struttura Packet che riguardano il livello datalink chiamando poi, prima di ritornare, la routine di decodifica appropriata per il livello superiore network.

Le routine di decodifica del livello network lavorano quindi allo stesso modo; anch'esse riempiranno i campi appropriati della struttura Packet per poi passare il controllo, se necessario, alle routine di decodifica del livello di trasporto.

Quando la decodifica del pacchetto sarà terminata, il pacchetto, eventualmente registrato nei sistemi di log, passerà attraverso i diversi preprocessori, ad esempio scritti specificatamente per il protocollo HTTP o per

ISSN: 2279-8773

l'individuazione dei portscan, ed infine arriverà al detection engine.

Quest'ultimo scorrerà la lista linkata delle regole e confronterà il pacchetto con ognuna di esse: se quindi il pacchetto corrisponderà ad una delle regole presenti, il processo di controllo verrà interrotto e la specifica azione corrispondente, ad esempio di alert o di log, verrà intrapresa.

In ogni caso, ci sono una serie di linee guida che devono essere seguite nella scrittura delle regole di Snort: esse infatti devono essere scritte su di un'unica riga, ed è necessario suddividerle in due sezioni logiche:

- *header*, che contiene le azioni delle regole, il protocollo di riferimento, gli indirizzi IP sorgente e destinazione e le informazioni sulle porte: di conseguenza esso contiene le informazioni che definiscono "chi, dove e cosa fare" dei pacchetti;
- *options*, che contiene i messaggi di alert da generare e tutte le informazioni sulle parti dei pacchetti che devono essere analizzate per determinare se le regole di azione

ISSN: 2279-8773

debbano essere eseguite o meno: tale sezione non è comunque specificatamente richiesta da tutte le regole.

Una tipica regola di Snort può essere formalizzata come espresso di seguito:

```
action protocol source_ip source_port -> dest_ip
dest_port (opt1; opt2; ...; opt<n>)
```

In particolare è possibile notare:

- *action*, che indica l'azione della regola e può assumere i seguenti valori:
  - *alert*, che genera un alert usando il metodo di alert scelto e poi esegue il log del pacchetto;
  - *log*, che esegue esclusivamente il log del pacchetto;
  - *pass*, che ignora il pacchetto non generando alcun allarme;
  - *activate*, che genera un alert che poi attiva un'azione dinamica.

ISSN: 2279-8773

- *protocol*, che indica il protocollo utilizzato dalla connessione corrispondente al pacchetto analizzato, ticpicamente TCP, UDP o ICMP;
- *source_ip* e *dest_ip*, che indicano rispettivamente l'indirizzo IP sorgente e l'indirizzo IP destinazione del pacchetto insieme all'eventuale netmask utilizzata;
- *source_port* e *dest_port*, che indicano rispettivamente la porta sorgente e la porta di destinazione del pacchetto. Attraverso l'operatore `:' può di fatto anche essere specificato un intervallo di porte;
- ->, che indica la direzione del traffico, dalla sorgente alla destinazione, ma che può essere anche bi-direzionale mediante il simbolo '<>'.

Inoltre, per ciò che concerne gli indirizzi IP e le porte, è possibile utilizzare la parola chiave 'any' per indicare che verrà accettato qualsiasi valore in ingresso, piuttosto che il simbolo `!' per negare la condizione che poi verrà specificata di seguito all'operatore di negazione stesso.

Le opzioni invece possono essere combinate in qualsiasi modo per rilevare e classificare i pacchetti di interesse, e

ISSN: 2279-8773

sono processate usando un AND logico; di fatto quindi tutte le opzioni elencate nella regola devono risultare rispettate affinché sia generato l'alert.

Tra le opzioni più significative si ricordano:

- *content*, che può individuare un determinato pattern nel payload del pacchetto e può contenere un misto di dati ascii-text e dati binari, questi ultimi generalmente rappresentati mediante la codifica in esadecimale;
- *msg*, che può produrre un messaggio di alert, a video o nei file di log, definito dall'utente;
- *flags*, che può individuare particolari condizioni relativamente alle flag del protocollo TCP.

Fra i diversi modi con i quali è possibile mantenere aggiornato il database delle regole, si segnala Oinkmaster [25], una script perl che, previa registrazione gratuita sul sito di Snort, consente di aggiornare, anche più volte al giorno, l'insieme delle regole sulle quali l'Intrusion Prevention System si basa.

ISSN: 2279-8773

3.2 MySQL

MySQL [26] è un Database management system (DBMS) relazionale, ovvero un sistema software progettato per consentire la creazione e la manipolazione efficiente di database o di collezioni di dati strutturati, composto da un client con interfaccia a caratteri ed un server, entrambi disponibili sia per sistemi Unix-like che per sistemi Windows.

Dal 1996 supporta la maggior parte della sintassi standard SQL, mentre dalla versione 5.x in poi aderisce pienamente anche allo standard ANSI SQL-2003.

Possiede potenti interfacce per numerosi linguaggi, compresi driver ODBC, driver Java e driver specifici per diverse piattaforme come Mono e quindi .NET.

Il codice di MySQL viene sviluppato fin dal 1979 dalla ditta chiamata 'TcX ataconsult', adesso 'MySQL AB'; in realtà però è solo dal 1996 che viene distribuita una versione che supporta il linguaggio SQL, tra l'altro solo dopo aver in parte riutilizzato il codice di un altro DBMS conosciuto come mSQL, che a tutti gli effetti poi è andato a sostituire.

ISSN: 2279-8773

Tale codice è comunque ad oggi di proprietà della omonima società e viene distribuito attraverso diverse forme di licensing tra cui si ricorda la licenza GNU/GPL ed una licenza commerciale la quale comprende il supporto del prodotto.

In aggiunta, di fatto tutta la parte client è licenziata mediante GNU/LGPL e può quindi essere utilizzata all'interno di applicazioni commerciali.

Tra le innumerevoli applicazioni degne di nota, MySQL assolve il compito di DBMS nella LAMP development suite, acronimo di 'Linux + Apache + MySQL + PHP/Perl/Python', una delle più utilizzate ed installate su Internet per lo sviluppo di siti ed applicazioni web dinamiche.

Inoltre si ricorda che organi governativi di rilevanza mondiale come la NASA americana, al fine di gestire la sua enorme mole di contenuti multimediali e di dati raccolti, ha sostituito un DBMS del calibro di Oracle migrando verso MySQL in virtù di alcune valutazioni di natura economica ma soprattutto in termini di supporto, di compatibilità e di prestazioni riscontrate.

ISSN: 2279-8773

Fino a qualche anno fa lo sviluppo del programma era opera soprattutto dei suoi sviluppatori iniziali: David Axmark, Allan Larsson e Michael Widenius.

Quest'ultimo rimane il principale autore del codice, oltre che principale socio della società, e tuttora supervisiona il progetto, tra l'altro valutando e coordinando i notevoli contributi che pervengono dai numerosi volontari sparsi in tutto il mondo.

3.3 Base

BASE [27], acronimo di Basic Analysis and Security Engine project, è un motore di analisi PHP based utilizzato per la ricerca e la gestione di un database che raccoglie eventi di sicurezza registrati da vari programmi di audit quali ad esempio Snort.

I sistemi DBMS supportati da BASE sono MySQL, PostgreSQL e Microsoft SQL Server.

BASE offre quindi un'interfaccia Web di analisi, tipicamente accedibile attraverso un server web PHP enabled come ad esempio Apache [28], dove poter costruire query

parametrizzate in base ai più comuni indicatori come il tempo, gli indirizzi IP sorgente e destinazione, le porte, i protocolli, i tipi di attacchi e cosi via, come mostrato in fig.8.

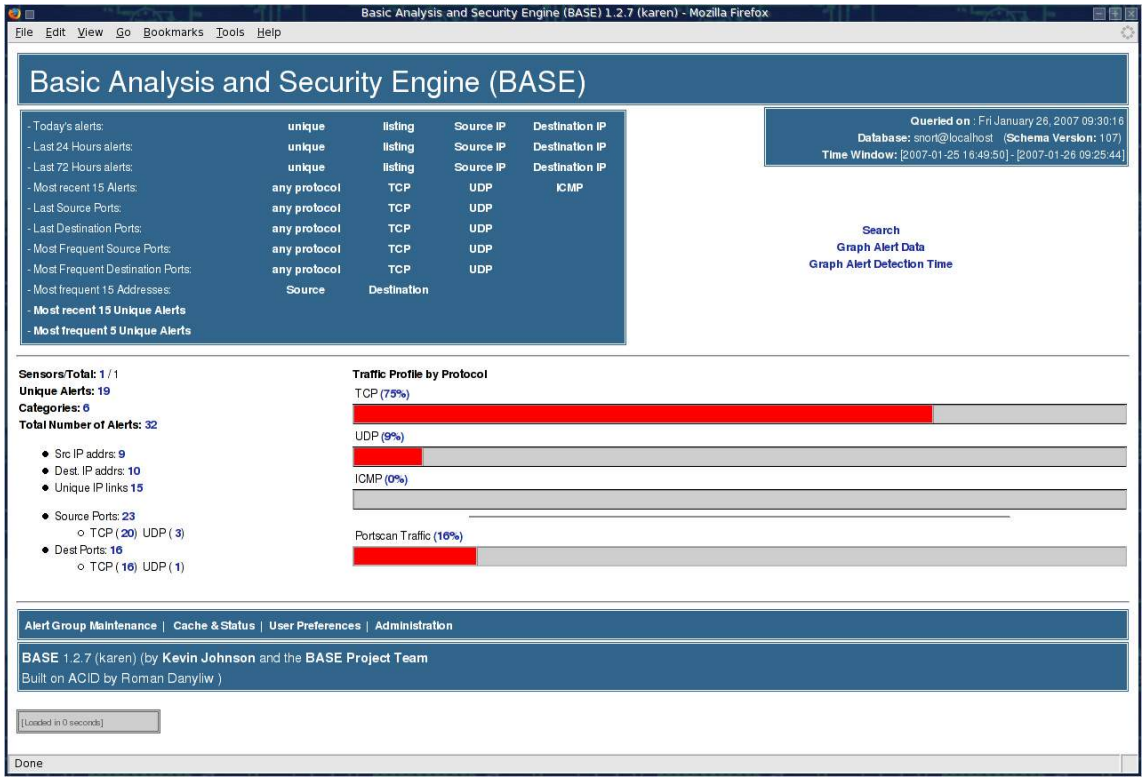

Fig. 8 - L'interfaccia di amministrazione di BASE.

Inoltre mediante BASE è possibile visualizzare e decodificare le informazioni a layer-3 e layer-4 dei pacchetti relativi ai singoli alert individuati.

ISSN: 2279–8773

Infine è possibile gestire direttamente le operazioni di manutenzione del database sottostante e quindi produrre grafici e statistiche relative alle diverse aggregazioni possibili dei dati raccolti.

3.4 Guardian

Guardian [29] è un programma di sicurezza che permette di eseguire una serie di comandi arbitrari, o vere e proprie script, in presenza di particolari condizioni rilevate da un Intrusion Detection System con il quale solitamente viene agganciato, quale ad esempio Snort.

In particolare, è possibile identificare un insieme di host da monitorare, e quindi solo per essi attivare una serie di procedure customizzate, oppure scegliere una lista di host da escludere, per i quali il programma di fatto non eseguirà alcuna azione.

Attraverso Guardian è quindi consentito invocare script arbitrarie che, come vedremo, ci torneranno molto utili nel presente lavoro di tesi.

ISSN: 2279-8773

Una particolare feature degna di nota è la durata dell'azione intrapresa: infatti, sia che essa sia un blocco, un messaggio remoto piuttosto che un comando locale, è possibile specificare un intervallo di tempo trascorso il quale verranno eseguite tutta una serie di operazioni di rilascio o di disattivazione delle misure dinamiche precedentemente attivate.

3.5 Gnokii

Gnokii [30] è un'applicazione Open Source che offre un insieme di tool e driver in user space per collegare un sistema Linux, Solaris, *BSD o Windows con un apparecchio conforme allo standard Global System for Mobile Communications (GSM), attraverso un'interfaccia seriale, un'interfaccia USB, mediante raggi infrarossi piuttosto che attraverso Bluetooth.

Tra le altre funzionalità, che sommariamente ricoprono tutte le feature di un apparecchio telefonico GSM, ad esempio dall'invio e ricezione delle chiamate alla gestione di appuntamenti e TODOlist, nel nostro caso si vedrà come

ISSN: 2279-8773

inviare brevi messaggi di testo (SMS) attraverso un telefono Nokia modello 3310, allo scopo di notificare ad uno o più destinatari il verificarsi di certi eventi o allarmi, piuttosto che l'esecuzione di determinate operazioni dinamiche eseguite dal sistema sottostante di prevenzione delle intrusioni al quale il dispositivo telefonico è agganciato.

3.6 Altri strumenti

Durante la fase sperimentale illustrata nel quarto ed ultimo capitolo si mostrerà, tra le altre cose, l'interazione di un IPS Snort-based con un firewall, con un router ed infine con un sistema GSM.

Per raggiungere tali obiettivi, oltre che il software Gnokii descritto nel precedente paragrafo, si sono utilizzati il celebre Firewall Iptables [31], principale componente della suite Netfilter [32], ed il tool conosciuto come Iproute2 [33], utilizzato per l'interazione con il sistema di routing a basso livello.

ISSN: 2279-8773

Netfilter è un componente dei sistemi operativi Linux-based che permette l'intercettazione e la manipolazione dei pacchetti che attraversano una Linux box.

Esso implementa diverse funzionalità di rete avanzate, permettendo quindi di realizzare firewall stateful anche molto particolari e complessi, ad esempio basati sul filtraggio dei pacchetti o su Network Address Translation (NAT).

Iptables è di fatto il programma che permette di configurare Netfilter, definendo le regole per i filtri di rete, il re-indirizzamento NAT e la manipolazione dei pacchetti stessi; spesso comunque con il termine Iptables ci si riferisce piuttosto all'intera infrastruttura che implementa le funzionalità di packet filtering e di packet mangling messe a disposizione dall'intera suite.

Iproute2 è invece una collezione di utilities per controllare il traffico e l'indirizzamento di una rete TCP/IP di un sistema Linux.

Oltre che meccanismi avanzati di routing, la suite permette di implementare particolari funzioni di rete quali ad

ISSN: 2279-8773

esempio Traffic Shaping e Quality of Service (QoS) [34] mediante la definizione di particolari classi, filtri e diverse discipline di incodamento.

3.7 Il Centro Elaborazione Dati (CED)

In quest'ultimo paragrafo si riporta l'architettura di rete realizzata presso il Centro elaborazione Dati (CED) della Ripartizione informatica dell'Università degli Studi di Perugia, ente per il quale lavoro dall'anno 2000.

Per motivi di privacy e sicurezza verranno omessi nomi e indirizzi IP reali; tale architettura, visibile in fig.9, in realtà servirà a comprendere la reale collocazione dell'Intrusion Prevention System all'interno della rete universitaria, le sue funzionalità ed il suo utilizzo principale previsto nella fase sperimentale.

In particolare si intenderà con la stringa '**svc net**' la rete di classe C il cui traffico in ingresso ed in uscita verrà analizzato e quindi gestito dall'IPS tramite la sua interfaccia di monitoring denominata '**monitor port**': in questa rete sono presenti i database server, gli application server ed i

ISSN: 2279-8773

servizi di clustering del CED.

Con la stringa '**admin net**' intendiamo invece un'altra rete di classe C suddivisa, attraverso una subnet /26, in quattro sottoreti di 64 host ognuna: tali sottoreti vengono utilizzate per scopi amministrativi, ad esempio dall'IPS stesso attraverso una seconda interfaccia denominata '**admin port**', o per altri scopi che esulato dal presente lavoro di tesi.

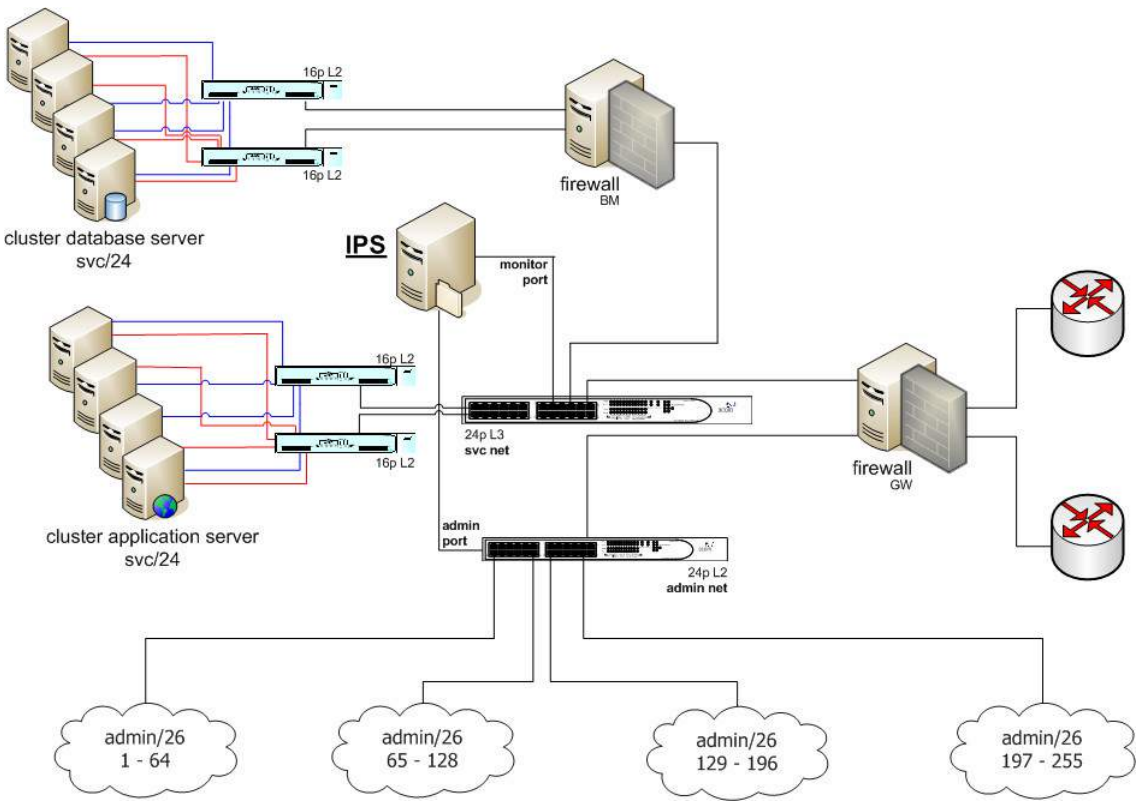

Fig. 9 – L'architettura del Centro Elaborazione Dati (CED).

ISSN: 2279-8773

In figura viene mostrata un'immagine reale dell'armadio di rete principale del CED stesso.

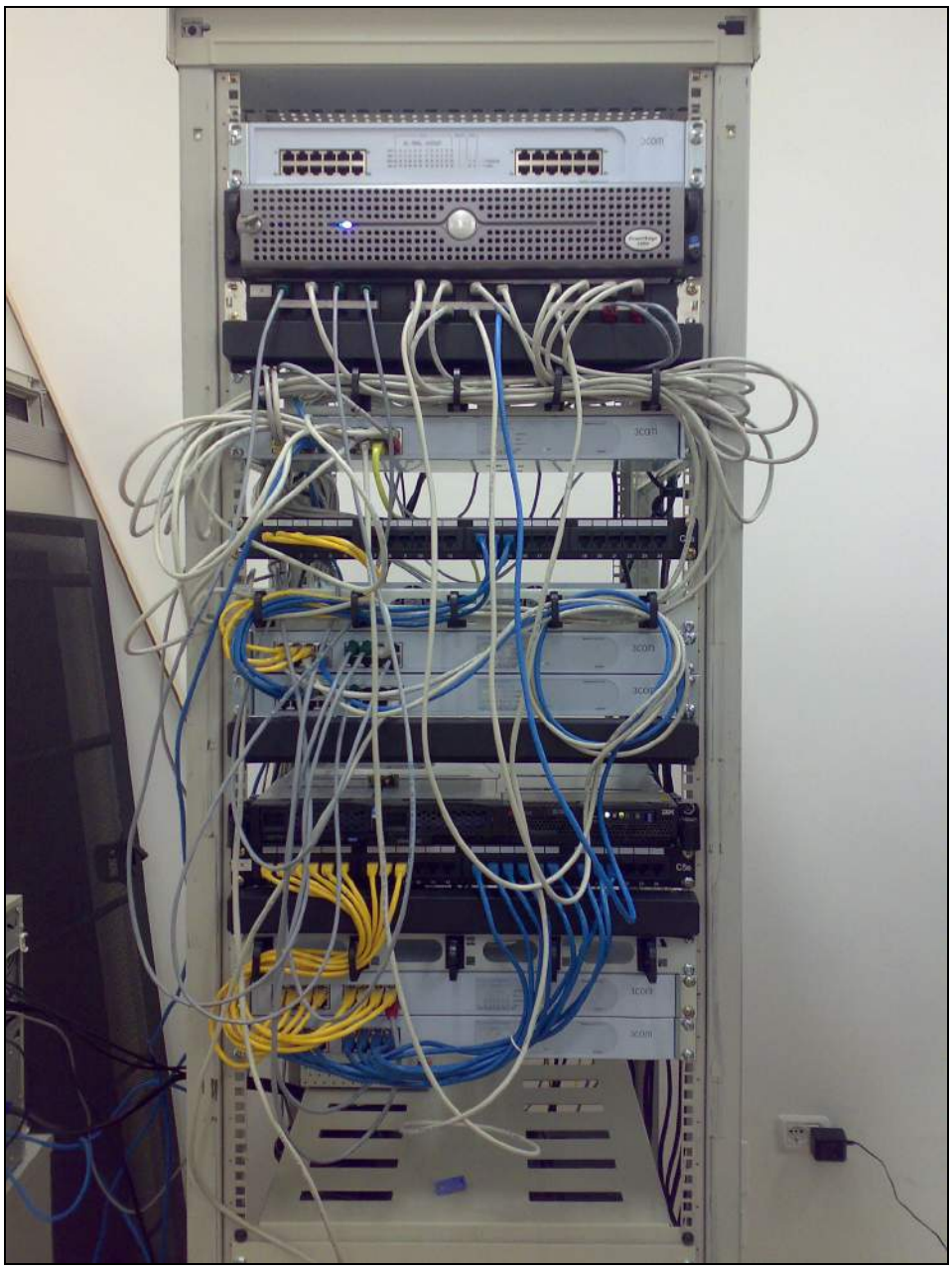

Fig. 10 – Dispositivi di rete nel Centro Elaborazione Dati.

ISSN: 2279-8773

Partendo dall'alto verso il basso di fig.10 è possibile notare il terzo dispositivo che corrisponde ad uno switch 3-Com 3824, componente principale del CED dove si attesta tutto il traffico in esame relativamente alla rete dei servizi '**svc net**'.

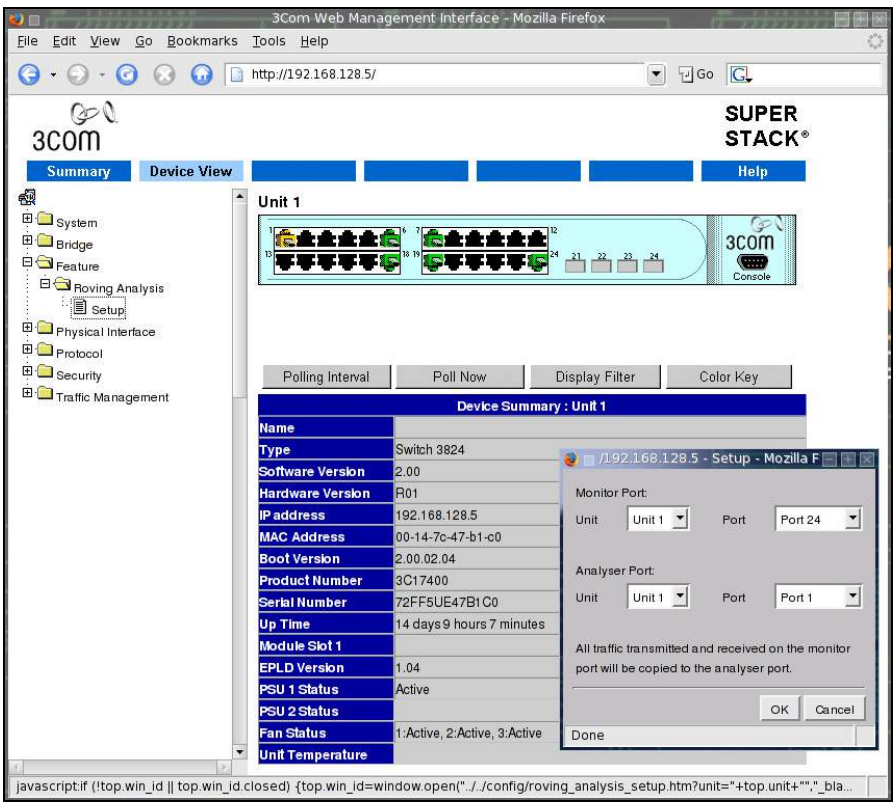

Fig. 11 – Pannello di configurazione dello switch 3-Com.

In fig.11 è infine mostrata l'interfaccia web di amministrazione attraverso la quale è stata configurata la

ISSN: 2279–8773

replicazione del traffico dalla porta n.1 dello switch alla porta n.24, ovvero la '**monitor port**' dell'Intrusion Prevention System.

ISSN: 2279-8773

Capitolo 4 -

Realizzazione del Progetto

Con il quarto ed ultimo capitolo è mia intenzione illustrare passo passo lo sviluppo del lavoro di tesi, illustrando quindi un mini-howto del sistema realizzato, dall'installazione e dalla configurazione fino alla personalizzazione ed alla fase di tuning, tutte descritte nei primi tre paragrafi.

Infine, prima di presentare la verifica del sistema, nel quinto ed ultimo paragrafo, si cercherà di stimolare una serie di spunti, di osservazioni e di riflessioni a fronte dei diversi contributi originali prodotti, descritti nel quarto paragrafo.

4.1 Installazione e configurazione

Il sistema è stato installato su piattaforma Linux server Intel/i386 con sistema operativo Mandriva 2007.0, kernel 2.6.17-10.

ISSN: 2279-8773

Nel nostro caso, al fine di installare il NIDS Snort [35] è stato necessario anche installare, per soddisfare i prerequisiti richiesti e le diverse compatibilità, le librerie Libnet [36], Libdnet [37] e Libpcap [38], come mostrato di seguito:

```
# urpmi libpcap
# urpmi libpcap-devel
# urpmi libnet1.0.2
# urpmi libnet1.0.2-devel
# urpmi libdnet1
# urpmi libdnet1-devel
```

Queste librerie servono sostanzialmente per interfacciare diversi protocolli, routine di rete di basso livello e per gestire il packet capturing.

A questo punto, prima di procedere con l'effettivo setup di Snort, si è installata la parte client e la parte server del database MySQL:

```
# urpmi MySQL-5.0.24a-2
# urpmi MySQL-client-5.0.24a-2
# urpmi MySQL-common-5.0.24a-2
```

Si è quindi pronti a scaricare il pacchetto dei sorgenti `snort-2.6.0.2.tar.gz`, quindi ad estrarlo, a configurarlo

ISSN: 2279-8773

(come mostrato nel dettaglio in appendice A), a compilarlo e ad installarlo come riportato mediante i seguenti comandi:

```
# wget http://www.snort.org/dl/current/snort-
2.6.0.2.tar.gz
# tar –zxvf snort-2.6.0.2.tar.gz
# cd snort-2.6.0.2
# ./configure --prefix=/usr/local \
--includedir=/usr/include/mysql \
--enable-dynamicplugin \
--enable-perfprofiling \
--enable-timestats \
--enable-flexresp2 \
--enable-react \
--enable-linux-smp-stats \
--with-snmp \
--with-openssl \
--with-mysql |tee configure.log
# make |tee make.log
# make install |tee make_install.log
[...]
Libraries have been installed in:
/usr/local/lib/snort_dynamicpreprocessor
/usr/local/lib/snort_dynamicengine
add LIBDIR to `/etc/ld.so.conf'
[...]
```

In particolare la compilazione avviene specificando la directory di installazione `/usr/local` e la directory dei file di 'include' di MySQL, abilitando le opzioni desiderate di Snort come il caricamento dinamico di plugin o particolari funzioni statistiche (per le specifiche dettagliate di ogni

ISSN: 2279-8773

opzione si può consultare il file INSTALL presente nella distribuzione sorgente), e quindi includendo il supporto per SNMP, SSL ed infine per il DBMS MySQL stesso.

La fase finale del 'make install' riporta l'installazione delle librerie dei preprocessori e del motore dinamico principale di Snort sulla directory `/usr/local/lib`, da aggiungere al file `/etc/ld.so.conf` del sistema.

Dopodiché, dopo avere eseguito il comando `ldconfig()` per aggiornare dinamicamente i binding delle librerie di sistema shared a run-time, si crea e si popola la directory di configurazione `/etc/snort` come mostrato attraverso i seguenti comandi:

```
# mkdir /etc/snort
# cd /etc/snort
# cp /src/snort-2.6.0.2/etc/snort.conf .
# cp /src/snort-2.6.0.2/etc/threshold.conf .
# cp /src/snort-2.6.0.2/etc/unicode.map .
# cp /src/snort-2.6.0.2/etc/sid-msg.map .
# cp /src/snort-2.6.0.2/etc/gen-msg.map .
# cp /src/snort-2.6.0.2/etc/reference.config .
# cp /src/snort-2.6.0.2/etc/classification.config
.
```

Quindi si scarica un primo file delle regole estraendolo su tale directory:

ISSN: 2279-8773

```
# wget http://www.snort.org/pub-
bin/downloads.cgi/Download/comm_rules/Community-
Rules-CURRENT.tar.gz
# tar -zxvf Community-Rules-CURRENT.tar.gz
# ls
snort.conf  threshold.conf  classification.config
reference.config     sid-msg.map     gen-msg.map
unicode.map  Community-Rules-CURRENT.tar.gz  rules/
docs/
```

Si apre quindi il file di configurazione principale di Snort `/etc/snort/snort.conf` e si modificano le seguenti variabili:

```
var eth1_ADDRESS [<ip monitor port>/32]
var HOME_NET [<svc net>/24,<admin net>/26]
var EXTERNAL_NET any
var RULE_PATH /etc/snort/rules
```

Tenendo a mente la fig.9 del capitolo 3, che riporta l'infrastruttura del Centro Elaborazione Dati (CED) su cui si poggia la sperimentazione, si specifica l'interfaccia 'eth1' come monitor port, gli equivalenti numerici delle subnet 'svc' ed 'admin' come reti locali e, tramite la parola chiave 'any', come reti esterne tutte le altre reti; infine si specifica la directory `/etc/snort/rules` che contiene le regole dell'IPS.

ISSN: 2279-8773

Inoltre, per ogni regola presente nella directory `$RULE_PATH` sarà necessario inserire nel file di configurazione `/etc/snort/snort.conf` delle direttive del tipo:

```
include $RULE_PATH/spyware-put.rules
include $RULE_PATH/specific-threats.rules
include $RULE_PATH/unipg.rules
[...]
```

Esse indicano che dovranno essere considerate tutte le regole specificate nei rispettivi file indicati.

A questo punto si eseguiranno un'ulteriore serie di operazioni che ci condurranno ad un primo test di funzionamento dell'IPS:

```
# mkdir /var/log/snort
# cp /src/snort-2.6.0.2/rpm/snort.logrotate
/etc/logrotate.d/snort
# cp /src/snort-2.6.0.2/rpm/snort.sysconfig
/etc/sysconfig/snort
# cp /src/snort-2.6.0.2/rpm/snortd /etc/init.d/
# cd /etc/rc3.d; ln -s ../init.d/snortd S89snortd
```

Attraverso questa serie di semplici comandi rispettivamente si crea la directory `/var/log/snort` per i file di log, si specificano il file `/etc/logrotate.d/snort` per la loro

ISSN: 2279-8773

rotazione, il file di configurazione globale `/etc/sysconfig/snort` letto dallo script di avvio del demone `/etc/init.d/snortd` ed il suo relativo link simbolico creato in `/etc/rc3.d` per far partire l'IPS a runlevel 3.

Tra le diverse impostazioni possibili, nel nostro caso sarà necessario impostare le seguenti variabili nel file `/etc/sysconfig/snort`:

```
INTERFACE=eth1
USER=root
GROUP=root
BPFFILE=/etc/snort/excludes-dpp.conf
```

In particolare si specifica l'interfaccia sulla quale far partire il sensore, l'utente ed il gruppo utilizzati per l'esecuzione del demone, ed un file, chiamato `/etc/snort/excludes-dpp.conf`, utilizzato per filtrare a priori le diverse azioni dell'IPS secondo la sintassi del noto programma `tcpdump()`.

In pratica tale file, che verrà analizzato nel terzo paragrafo relativamente alla fase di tuning del sistema, viene consultato prima dell'inizializzazione dei diversi

ISSN: 2279-8773

preprocessori, e quindi di fatto prima che il demone parta.

A questo punto tutto è pronto; basta avviare Snort con il comando:

```
# /etc/init.d/snortd start
```

ed analizzare il file di log principale del sistema `/var/log/messages` per accorgersi che l'IPS è stato avviato con successo (l'output completo restituito dal demone all'avvio è riportato in appendice B):

```
Running in IDS mode
        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file /etc/snort/snort.conf
Initializing rule chains..
Log directory = /var/log/snort
Loading dynamic engine
/usr/local/lib/snort_dynamicengine/libsf_engine.so
... done
Loading all dynamic preprocessor libs from
/usr/local/lib/snort_dynamicpreprocessor/...
Finished Loading all dynamic preprocessor libs
from /usr/local/lib/snort_dynamicpreprocessor/
Initializing Network Interface eth1
Decoding Ethernet on interface eth1
        --== Initialization Complete ==--
```

Si andrà quindi a verificare, mediante il comando `ps()`, che

ISSN: 2279-8773

tutto stia funzionando correttamente:

```
# ps -ef |grep snort
root      10082      1 13 08:57 ?          00:00:10
/usr/local/bin/snort -b -d -k none -D -i eth1 -u
root   -g   root   -c   /etc/snort/snort.conf   -l
/var/log/snort -F /etc/snort/excludes-dpp.conf
```

4.2 Personalizzazione

Si illustrerà ora la personalizzazione dell'ambiente; per prima cosa si desidera che tutte le operazioni di logging e di alerting vengano effettuate su database MySQL.

Allo scopo si crea il database principale chiamato 'snort' in questo modo:

```
# echo "CREATE DATABASE snort;" | mysql -u root –p
```

A questo punto si creano le tabelle necessarie, secondo lo schema fornito con la distribuzione sorgente, e quindi si assegnano all'utente chiamato 'sensor' i giusti permessi per accedere ad esse esclusivamente attraverso l'interfaccia di localhost:

```
# mysql -D snort -u root -p < /src/snort-
2.6.0.2/schemas/create_mysql
# mysql -u root -D mysql -p
```


ISSN: 2279–8773

```
mysql> GRANT INSERT, SELECT on snort.* to
sensor@localhost IDENTIFIED BY 'password';
mysql> quit
```

Quindi si modifica la plugin di output di snort in modo tale da utilizzare, invece che il consueto sistema di logging su file, il database MySQL appena installato, specificando in `/etc/snort/snort.conf` la direttiva:

```
output database: alert, mysql, user=sensor
password=password dbname=snort host=localhost
```

Infine, prima di riavviare il demone `snortd`, sarà necessario commentare l'opzione '`ALERTMODE=fast`' in `/etc/sysconfig/snort`, abilitata di default, che altrimenti andrebbe a sovrascrivere l'output database plugin precedentemente specificata.

A questo punto verrà installata l'interfaccia di amministrazione BASE che ci permetterà di configurare, visualizzare e quindi di gestire il contenuto dell'intero database, dagli alert raccolti alla manutenzione delle tabelle stesse:

```
# cd /var/www/html
# tar -zxvf base-1.2.7.tar.gz
```


ISSN: 2279-8773

```
# install php-pear-*
# pear install numbers_words-beta
# pear install numbers_roman
# pear install image_color
# pear install image_canvas-alpha
# pear install image_graph-alpha
```

Una volta posizionatisi sulla root directory del server web, ad esempio Apache Web Server nel presente lavoro di tesi, sarà sufficiente decomprimere il pacchetto e quindi accedervi la prima volta per configurare l'intera console di amministrazione.

Nel nostro caso è stato inoltre necessario installare il comando 'pear', un'estensione PHP mediante la quale è possibile gestire una libreria strutturata di funzioni, aggiungendo quindi i diversi componenti necessari per un corretto funzionamento della console di gestione, come ad esempio la produzione, parametrizzata dall'utente, di statistiche o di grafici in tempo reale.

Quindi sarà necessario accedere nuovamente al database per aggiungere degli ulteriori privilegi all'utente di snort, in particolare i permessi di UPDATE, CREATE e DELETE per le tabelle a cui BASE dovrà accedere, come mostrato di seguito mediante i seguenti comandi:

ISSN: 2279-8773

```
# mysql -p
mysql>    GRANT    UPDATE    on    snort.*    to
sensor@localhost;
mysql>    GRANT    CREATE    on    snort.*    to
sensor@localhost;
mysql>    GRANT    DELETE    on    snort.*    to
sensor@localhost;
mysql> quit
```

L'insieme dei permessi dell'utente sensor può quindi essere visualizzato con la semplice query '`select * from db`' sul database di sistema di MySQL chiamato '`mysql`'.

Per verificare che tutto stia funzionando correttamente, è possibile utilizzare il comando:

```
# echo "SELECT count(*) FROM event" | mysql -D
snort -p
Enter password: ******
count(*)
145
```

In questo modo verrà restituito un numero che corrisponde agli alert registrati al momento, che ovviamente dovrà risultare diverso da zero o comunque crescente al verificarsi dei diversi eventi sulla rete.

A questo punto abbiamo ottenuto un sistema di Network Intrusion Detection perfettamente funzionante, ma che

ISSN: 2279-8773

ancora non prevede altre possibilità che una qualche registrazione degli eventi che si verificano; il nostro scopo, da ora in poi, sarà quindi quello di trovare un sistema per attivare una serie di reazioni in corrispondenza dei nostri target da proteggere.

Allo scopo si installa Guardian, il quale ci permetterà, monitorando a sua volta gli alert individuati da Snort, di invocare una serie di comandi arbitrari in corrispondenza di particolari eventi.

L'installazione viene realizzata tramite i comandi:

```
# cd /src && tar -zxvf guardian-1.7.tar.gz
# touch /etc/guardian.target \
/etc/guardian.ignore \
/var/log/guardian.log \
/usr/local/bin/guardian_block.sh \
/usr/local/bin/guardian_unblock.sh \
# chmod 750 /usr/local/bin/guardian_*.sh
# cp /src/guardian-1.7/guardian.pl /usr/local/bin/
# cp /src/guardian-1.7/guardian.conf /etc
```

In particolare, dopo avere estratto il pacchetto, si creeranno i file `guardian.target`, `guardian.ignore` e `guardian.log` dove rispettivamente si inseriranno la lista degli IP da considerare, la lista degli IP da escludere e dove verranno raccolti i log del programma.

ISSN: 2279-8773

Inoltre si creano due script vuote che verranno utilizzate più avanti per avviare le nostre azioni dinamiche customizzate, in particolare durante le fasi di blocco e di rilascio di un indirizzo ip attaccante.

Infine vengono copiati sulle directory corrette il demone `guardian.pl` ed il suo file di configurazione `guardian.conf`.

Si modifica quest'ultimo file come segue:

```
HostIpAddr      <ip address snort sensor>
Interface       eth1
AlertFile       /var/log/snort/alert
TimeLimit       86400
```

In questo caso verranno specificati l'indirizzo IP del sensore da considerare, l'interfaccia di monitoring, il file di log di default di snort nonchè un tempo, espresso in secondi, dopo il quale tutti gli IP bloccati verranno rilasciati; nel nostro caso il tempo di `86400` secondi corrisponde ad un'intera giornata.

A questo punto è possibile far partire Guardian, da linea di comando, come segue:

ISSN: 2279-8773

```
# /usr/local/bin/guardian.pl
OS shows Linux
My ip address and interface are:
<ip address snort sensor> eth1
Loaded 0 addresses from /etc/guardian.ignore
Loaded 29 addresses from /etc/guardian.target
Becoming a daemon..
```

Quindi, dopo aver verificato l'effettivo funzionamento in tempo reale, sarà possibile inserire la precedente istruzione in `/etc/rc.local` al fine di far partire il servizio all'avvio del sistema.

Si prenda ora in esame la seguente porzione di codice del demone `guardian.pl`:

```
if (-x "/usr/local/bin/guardian_block.sh") {
  $blockpath = \
  "/usr/local/bin/guardian_block.sh";
}
if (-x "/usr/local/bin/guardian_unblock.sh") {
  $unblockpath = \
  "/usr/local/bin/guardian_unblock.sh";
}
```

Tale codice viene mostrato in quanto, dopo piccole modifiche che hanno riguardato le modalità con le quali vengono cercati i diversi programmi nel path di sistema, è qui che viene effettuato il collegamento tra il demone stesso e le due script vuote create in precedenza, le quali

ISSN: 2279-8773

verranno invocate rispettivamente ad ogni generazione di un alert di snort che coinvolga come vittima uno dei target presenti in `/etc/guardian.target`, ed in seguito al superamento del tempo di blocking per l'ip attaccante corrispondente.

Le due script, `guardian_block.sh` e `guardian_unblock.sh`, verranno presentate nel quarto paragrafo, relativamente ai contributi originali prodotti.

Prima di concludere, si è installato Gnokii al fine di inviare sms tramite un dispositivo GSM, nel nostro caso un telefono Nokia modello 3310.

Dopo aver collegato il telefono attraverso un cavo seriale, acceso il dispositivo ed installato i pacchetti necessari in questo modo:

```
# urpmi gnokii-0.6.4-4
# urpmi libgnokii3-0.6.13-6
# urpmi libgnokii2-0.6.8-2
```

si può aprire il file di configurazione chiamato `/etc/gnokiirc` e quindi definire le seguenti variabili, presenti principalmente in tre diverse sezioni:

ISSN: 2279-8773

```
[global]
port = /dev/ttyS0
model = 3310
initlength = default
connection = serial
use_locking = yes
serial_baudrate = 19200
smsc_timeout = 10

[connect_script]
TELEPHONE = +393476448035

[logging]
debug = on
rlpdebug = off
xdebug = off
```

In particolare si specifica la porta seriale `dev/ttyS0` alla quale è connesso il dispositivo GSM, il modello 3310, il tipo di collegamento `serial`, il numero `+393476448035` del telefono ed infine le impostazioni per il logging delle operazioni.

A questo punto basterà verificare il riconoscimento del dispositivo tramite il comando ‘`gnokii --identify`’ il cui output, insieme all’invio di un messaggio di test corredato da uno screenshot d’esempio, viene riportato in Appendice C.

A conclusione del paragrafo, in fig.12 viene mostrato il

ISSN: 2279-8773

telefono Nokia, modello 3310, collegato all'Intrusion Prevention System realizzato.

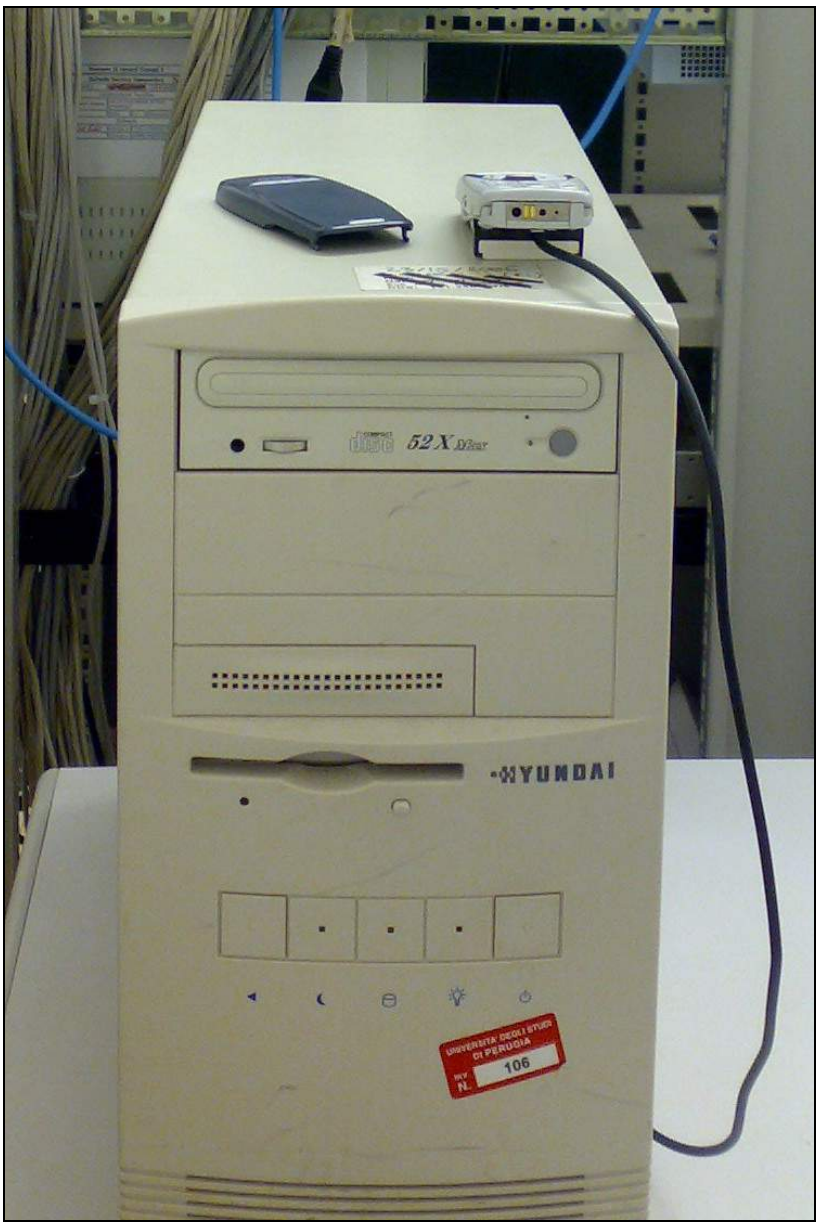

Fig. 12 – Il Nokia 3310 collegato al server Linux AMD/i386.

ISSN: 2279–8773

4.3 Tuning

Durante questo paragrafo si descrive prima il problema dei falsi positivi e dei falsi negativi, quindi si presenteranno le metodologie per l'aggiornamento e la scrittura di nuove regole.

Si intende con falso positivo un alert generato dal sistema anche quando in realtà l'attività che ha scatenato l'avvio di tale evento è di fatto un'attività perfettamente consentita; per affrontare questo fastidioso problema si è pensato di scrivere un file ad-hoc di regole di 'pass', che poi verrà aggiornato ogni volta che si verificheranno nuovi eventi corrispondenti a falsi positivi.

Dopo un breve tempo che potremmo definire di 'apprendimento manuale' è stato possibile eliminare quasi totalmente questo tipo di messaggi fuorvianti; in particolare il file di regole, chiamato appunto `pass.rules`, contiene entry del tipo:

```
# attack-responses.rules
# <backup_server>.unipg.it
pass tcp 141.250.0.0/16 any -> <ip_backup_server>
1500 (msg:"ATTACK-RESPONSES directory listing";
```


ISSN: 2279-8773

```
flow:established; content:"Volume Serial Number";
classtype:bad-unknown; sid:1292; rev:9;)
# rpc.rules
# <backup_server>.unipg.it
pass udp <ip_backup_server> any -> $HOME_NET 111
(msg:"RPC portmap proxy attempt UDP"; content:"|00
01 86 A0|"; depth:4; offset:12; content:"|00 00 00
05|"; within:4; distance:4; content:"|00 00 00
00|"; depth:4; offset:4; classtype:rpc-portmap-
decode; sid:1923; rev:6;)
# icmp-info.rules
# <dns_server>.unipg.it
pass icmp <ip_dns_server> any -> $EXTERNAL_NET any
(msg:"ICMP Destination Unreachable Port
Unreachable"; icode:3; itype:3; classtype:misc-
activity; sid:402; rev:7;)
```

Per ogni entry si riporta il file originale che aveva generato l'alert, l'host per il quale si vuole invece consentire il tipo di traffico individuato (evidenziati in corsivo), e quindi la regola specifica dove si utilizzano i consueti parametri per identificare il protocollo (nell'esempio si mostra una regola rispettivamente per i protocolli TCP, UDP e ICMP), l'indirizzo ip coinvolto (oppure una subnet) e le porte sorgente e destinazione.

Si noti inoltre l'utilizzo delle variabili `$HOME_NET`, `$EXTERNAL_NET` e delle parole chiave 'pass' e 'any', che consentono di distinguere in maniera molto semplice l'oggetto di riferimento di ogni regola.

ISSN: 2279-8773

L'inclusione di questo file customizzato viene effettuata specificando in `/etc/snort/snort.conf` la direttiva:

```
include $RULE_PATH/pass.rules
```

Tale comando deve ovviamente essere posto in testa alle successive direttive di 'include' che poi specificheranno tutte le altre regole che invece definiscono gli alert.

In realtà però Snort utilizza, per le applicazioni più note, diversi preprocessori che ispezionano il traffico prima che esso arrivi ad essere confrontato con queste regole, di fatto producendo, se necessario, degli allarmi per proprio conto [39].

Nel nostro caso si è previsto che il preprocessore chiamato 'http_inspect_server' esegua i propri controlli anche sulla porta 8888 oltrechè sulla consueta 80 mentre il preprocessore 'sfportscan', responsabile dell'individuazione dei portscan, ignori alcuni ip sorgente o destinazione:

```
preprocessor http_inspect_server:
     server default \
     profile all ports { 80 8888 } \
     oversize_dir_length 500

preprocessor sfportscan:
```


ISSN: 2279-8773

```
proto  { all } \
scan_type { all } \
logfile { unipg-portscan.log } \
memcap { 10000000 } \
ignore_scanners { <ip_allow_to_do_pscan> } \
ignore_scanned { <ip_allow_to_recv_pscan> } \
sense_level { medium }
```

Questi preprocessori, fra cui si ricordano anche `http_inspect`, `ftp_telnet`, `smtp` e `dns`, eseguono dei controlli relativi ad attacchi particolari e specifici, rispettivamente per i protocolli HTTP, FTP, TELNET, SMTP e DNS, prevedendo, come nel caso del preprocessore 'sfportscan', anche un sistema di logging separato.

Di fatto, per filtrare anche l'attività di questi preprocessori, si è utilizzato il file `/etc/snort/excludes-dpp.conf`, di cui si è già accennato nel primo paragrafo.

In questo file sono quindi specificate una serie di espressioni del tipo:

```
not (dst host 192.168.198.135 and src portrange
1024-65535 and dst port 8888)
and
not (src host 192.168.1.141 and src port 21 and
dst host 192.168.198.2 and dst portrange 1024-
65535)
```

Ad esempio in questo caso si esclude dal controllo dei

ISSN: 2279-8773

preprocessori il traffico diretto sulla porta 8888 dell'host 192.168.198.135 in quanto esso risulta essere il proxy server della rete locale per il quale molte delle segnalazioni prodotte dal preprocessore http_inspect non sono da ritenersi valide.

Per ulteriori approfondimenti, e per rendere l'idea dell'attività necessaria da svolgere, e quindi eseguita nel nostro contesto reale, si riporta in appendice D la lista dei file `excludes-dpp.conf` e `pass.rules`.

Un'altra problematica da affrontare riguarda poi l'aggiornamento delle regole, un'attività di non poco conto visto il presentarsi quotidiano di nuove forme di attacchi che potrebbero non essere sempre così facilmente individuati.

Nel successivo paragrafo si mostreranno le script prodotte per effettuare tre modalità differenti di aggiornamento basate su:

- regole rese disponibili dalla Community GPL;
- regole presenti sul tree CURRENT dello sviluppo via CVS;

ISSN: 2279-8773

- regole 'Sourcefire VRT Certified', disponibili per gli utenti gratuitamente registrati sul sito di Snort, rilasciate sotto VRT Certified Rules License [40].

Infine, per definire eventuali nuove regole, in relazione ad applicazioni proprie interne o anche semplicemente allo scopo di test, si è utilizzato il file chiamato `unipg.rules` così fatto:

```
# (C) Copyright 2007, Carlo Manuali
#    All rights reserved.
# $Id: unipg.rules,v 1.00 2007/02/01 15:46:11 bmc
Exp $
#------------
# UNIPG RULES
#------------
alert tcp any any -> any 2222 (flags:S; msg:"Test
Attack Connection";)
alert tcp any any -> any 80 (msg:"Web Attack";
uricontent:"/bin/wget"; nocase; sid:1025; rev:1;
classtype:web-application-attack;)
alert tcp any any -> any 22 \
    (\
        msg: "BETA Vulnerable SSH-1 Connection" ;\
        flags: PA ;\
        content: "SSH-1" ;\
        content: !".99-OpenSSH_3.8" ;\
        content: !"SSH-1.2-" ;\
    )
```

In questi tre casi d'esempio, con la prima regola vengono individuate tutte le connessioni verso la porta 2222, con la seconda regola tutto il traffico HTTP contenente sull'URI di

ISSN: 2279-8773

navigazione la stringa '`/bin/wget`' e con la terza regola tutte le connessioni client SSH-1 nelle versioni non ritenute affidabili.

In particolare si noti, in quest'ultima regola, la combinazione tra l'inclusione e l'esclusione di contenuti multipli tramite la parole chiave '`content`' e l'operatore di negazione '`!`'.

4.4 Contributi originali

In questo paragrafo si presentano le script principali prodotte in relazione allo sviluppo del presente lavoro di tesi.

Due script sicuramente significative e che caratterizzano la maggior parte di questo lavoro sono le script `guardian_block.sh` e `guardian_unblock.sh` che, invocate da Guardian, rispettivamente permettono la reazione dell'IPS agli attacchi ed il rilascio delle misure intraprese dopo l'intervallo di tempo definito.

La prima di esse è mostrata di seguito:

ISSN: 2279-8773

```
#!/bin/sh
source=$1
interface=$2
# FIREWALL
ssh root@<firewall>.unipg.it "iptables -I INPUT -s
$source -j DROP"
ssh root@<firewall>.unipg.it "iptables -I FORWARD
-s $source -j DROP"
# ROUTER
ssh root@<router>.unipg.it "ip route add $source
dev lo table local"
# SMS
cell_admin="+393395020217"
message_center="+393359609600"
echo "$source blocked by `hostname` int
$interface" | gnokii --sendsms $cell_admin --smsc
$message_center
# MAIL
rcpt_mail="carlo.manuali@unipg.it"
echo "$source blocked" |mail -s "snort attach
detected by `hostname` int "$interface $rcpt_mail
# PRINTER
printer_name="KYOCFS9120DN"
echo "$source blocked by `hostname` int
$interface" |lpr -P$printer_name
```

Dopo aver recuperato l'indirizzo ip dell'attaccante in `$source` e l'interfaccia di rilevamento del sensore in `$interface` si effettuano le seguenti operazioni:

- *collegamento al firewall di frontiera*, nel quale vengono aggiunte due regole di DROP per le chain di INPUT e di FORWARD relativamente all'indirizzo ip dell'attaccante;
- *collegamento al router di default*, per il quale viene aggiunta una null route verso il localhost per i pacchetti

ISSN: 2279-8773

destinati all'indirizzo ip dell'attaccante;

- *collegamento al dispositivo GSM*, ed invio di un messaggio verso un numero definito da programma, che riporta le azioni intraprese;
- *collegamento al sistema di posta elettronica*, ed invio di un messaggio all'amministratore che notifica l'evento di 'block' eseguito dell'IPS;
- *collegamento ad un dispositivo esterno*, quale una stampante locale dove viene nuovamente registrato l'evento in modo tale da non poter essere cancellato in nessun caso, anche a fronte di eventuali break dell'Intrusion Prevention System.

Successivamente, allo scadere del tempo stabilito in 24 ore a partire da ogni reazione dinamica attivata nei confronti di ogni singolo ip attaccante, verranno eseguiti i comandi presenti nella seconda script, mostrata di seguito, che di fatto 'liberano' l'indirizzo ip precedentemente bloccato:

```
#!/bin/sh
source=$1
interface=$2
# FIREWALL
ssh root@<firewall>.unipg.it "iptables -D INPUT -s
$source -j DROP"
```

```
ssh root@<firewall>.unipg.it "iptables -D FORWARD
-s $source -j DROP"
# ROUTER
ssh root@<router>.unipg.it "ip route del $source
dev lo table local"
# MAIL
rcpt_mail="carlo.manuali@unipg.it"
echo "$source released" |mail -s "snort expiration
made by `hostname` int "$interface $rcpt_mail
```

In particolare si effettuano le seguenti operazioni:

- *collegamento al firewall di frontiera*, nel quale vengono rimosse le due regole precedentemente inserite per le chain di INPUT e di FORWARD relativamente all'indirizzo ip dell'attaccante;
- *collegamento al router di default*, per il quale viene eliminata la null route verso il localhost precedentemente inserita per i pacchetti destinati all'indirizzo ip dell'attaccante;
- *collegamento al sistema di posta elettronica*, ed invio di un messaggio all'amministratore che notifica l'evento di 'unblock' eseguito dell'IPS.

Nel caso di rilascio dell'indirizzo ip dell'attaccante, si è scelto di non prevedere la notifica di tale azione né mediante GSM né tramite log sulla stampante per non

ISSN: 2279-8773

appesantire troppo il sistema.

Una seconda parte di script riguardano invece la manutenzione del sistema per ciò che concerne il controllo del RAID, lo stato dei filesytem e, più specificatamente in relazione al sistema di IPS, le script che realizzano le tre modalità di aggiornamento delle regole di Snort descritte nel precedente paragrafo.

Tutte le script menzionate vengono invocate attraverso `crontab()` in questo modo:

```
# Check Raid status every hour.
00 * * * * \
/usr/local/bin/chk_raid.sh
# 6.00 AM: Check Filesystems occupancy.
00 6 * * * \
/usr/local/bin/chk_filesystem.sh
# 6.30 AM: snort rules update - development tree.
30 6 * * * \
/usr/local/bin/upd_snort_rules-development.sh
# 7.00 AM: snort rules update - community tree.
00 7 * * * \
/usr/local/bin/upd_snort_rules-community.sh
# 7.30 AM: snort rules update - VRT current
(registered user release) tree.
30 7 * * * \
/usr/local/bin/upd_snort_rules-vrtcurrent.sh
```

In particolare la terza script, denominata `snort_rules-development.sh`, si collega, attraverso Concurrent Versions

ISSN: 2279-8773

System (CVS) [41], al tree di sviluppo di Snort scaricando le ultime regole disponibili.

Nel caso in cui siano presenti nuovi aggiornamenti, viene notificato l'amministratore di sistema e quindi vengono copiate le nuove regole sulla directory di produzione dell'IPS, come mostrato di seguito.

```
#!/bin/bash
# Define a local or remote admin e-mail address.
rcpt_mail="carlo.manuali@unipg.it"
# Define Snort CVS server.
server="cvs.snort.org:/cvsroot"
# Do it just first time.
# cvs -d:pserver:anonymous@$server login
# Define initial and final directories.
dwl_dir="/etc/snort/CVS_rules"
rules_dir="snort/rules"
# Main program.
cd $dwl_dir
output=`cvs -q -d:pserver:anonymous@$server \
co $rules_dir`
if [ -n "$output" ]; then
  echo $output |mail -s "snort rules \
  (development) updated for `hostname`" $rcpt_mail
fi
cp -pf $dwl_dir/$rules_dir/*.rules /etc/$rules_dir
```

La quarta script, denominata `snort_rules-community`, scarica invece attraverso il comando `wget()` l'insieme delle regole rilasciate dalla comunità GPL nella loro versione CURRENT.

ISSN: 2279-8773

Anche in questo caso, in presenza di nuovi aggiornamenti disponibili, viene notificato l'amministratore di sistema e quindi estratte ed inserite le nuove regole sulla directory di produzione dell'IPS, come mostrato di seguito:

```
#!/bin/bash
# Define a local or remote admin e-mail address.
rcpt_mail="carlo.manuali@unipg.it"
# Define initial and final directories.
dwl_dir="/etc/snort/CVS_rules"
rules_dir="/etc/snort/rules"
conf_dir="/etc/snort"
extr_dir="rules"
# Define URI to download.
protocol="http"
site="www.snort.org"
directory= \
"pub-bin/downloads.cgi/Download/comm_rules"
file="Community-Rules-CURRENT.tar.gz"
# Main Program.
cd $dwl_dir
wget $protocol://$site/$directory/$file \
>/dev/null 2>&1
if [ $? = 0 ]; then
  diff $dwl_dir/$file $conf_dir/$file 1>/dev/null
  if [ $? != 0 ]; then
    tar -zxvf $dwl_dir/$file $extr_dir 1>/dev/null
    mv -f $dwl_dir/$extr_dir/*.$extr_dir \
    $rules_dir
    chown root.root \
    $rules_dir/community-*.$extr_dir
    echo `date` |mail -s "snort rules \
    (community) updated for `hostname`" $rcpt_mail
    rm -rf $dwl_dir/$extr_dir
  fi
  mv -f $dwl_dir/$file $conf_dir/$file
fi
```


ISSN: 2279-8773

La quinta ed ultima script, denominata `snort_rules-vrtcurrent.sh`, infine recupera le regole aggiornate di Snort rilasciate dal Sourcefire Vulnerability Research Team (VRT) [42] per gli utenti (gratuitamente) registrati.

Di nuovo, in presenza di nuovi aggiornamenti, viene notificato l'amministratore di sistema circa l'update delle regole effettuato.

```
#!/bin/bash
# Recupera le regole aggiornate di Snort - VRT
# current (registered user release) Rules.
# Define a local or remote admin e-mail address.
rcpt_mail="carlo.manuali@unipg.it"
# Define rules directory.
rules_dir="/etc/snort/rules"
# Create a temp file.
tmp_file=`mktemp /tmp/oinkmaster.XXXXXXXX`
# use Oinkmaster program to do update, based on
# /etc/oinkmaster.conf.
/usr/local/bin/oinkmaster.pl -o $rules_dir -q > \
$tmp_file 2>&1
if [ -s $tmp_file ]; then
  mail -s "snort rules (VRT current) \
  updated for `hostname`" $rcpt_mail < $tmp_file
fi
# Delete temp file.
rm $tmp_file
```

Si noti che le prime due script, non essendo direttamente connesse con l'oggetto del presente lavoro di tesi, sono state semplicemente riportate per completezza in

ISSN: 2279-8773

Appendice E insieme ad alcune note sull'installazione, la configurazione e la personalizzazione del programma Oinkmaster utilizzato in quest'ultima script per gli aggiornamenti delle regole.

Un ultimo contributo riguarda un bug della versione di Snort `2.6.0.2` installata; in particolare è stato necessario aggiungere, sulla script di partenza del demone `/etc/init.d/snortd`, l'opzione '`-k none`' come mostrato di seguito:

```
daemon /usr/local/bin/snort $ALERTMODE $BINARY_LOG
$NO_PACKET_LOG     $DUMP_APP     -k     none     -D
$PRINT_INTERFACE $INTERFACE -u $USER -g $GROUP
$CONF -l $LOGDIR $PASS_FIRST $BPFFILE $BPF
```

Tale modifica si è resa necessaria al fine di disabilitare il controllo interno di verifica sulle checksum dei pacchetti, che non funzionava correttamente, di fatto invalidando senza alcun motivo l'invio delle notifiche di Snort configurato per essere eseguito in alert-mode.

ISSN: 2279-8773

4.5 Verifica ed esempi di funzionamento

In quest'ultimo paragrafo si illustrano due esempi di funzionamento e di verifica del sistema di Intrusion Prevention System (IPS) realizzato nel presente lavoro di tesi.

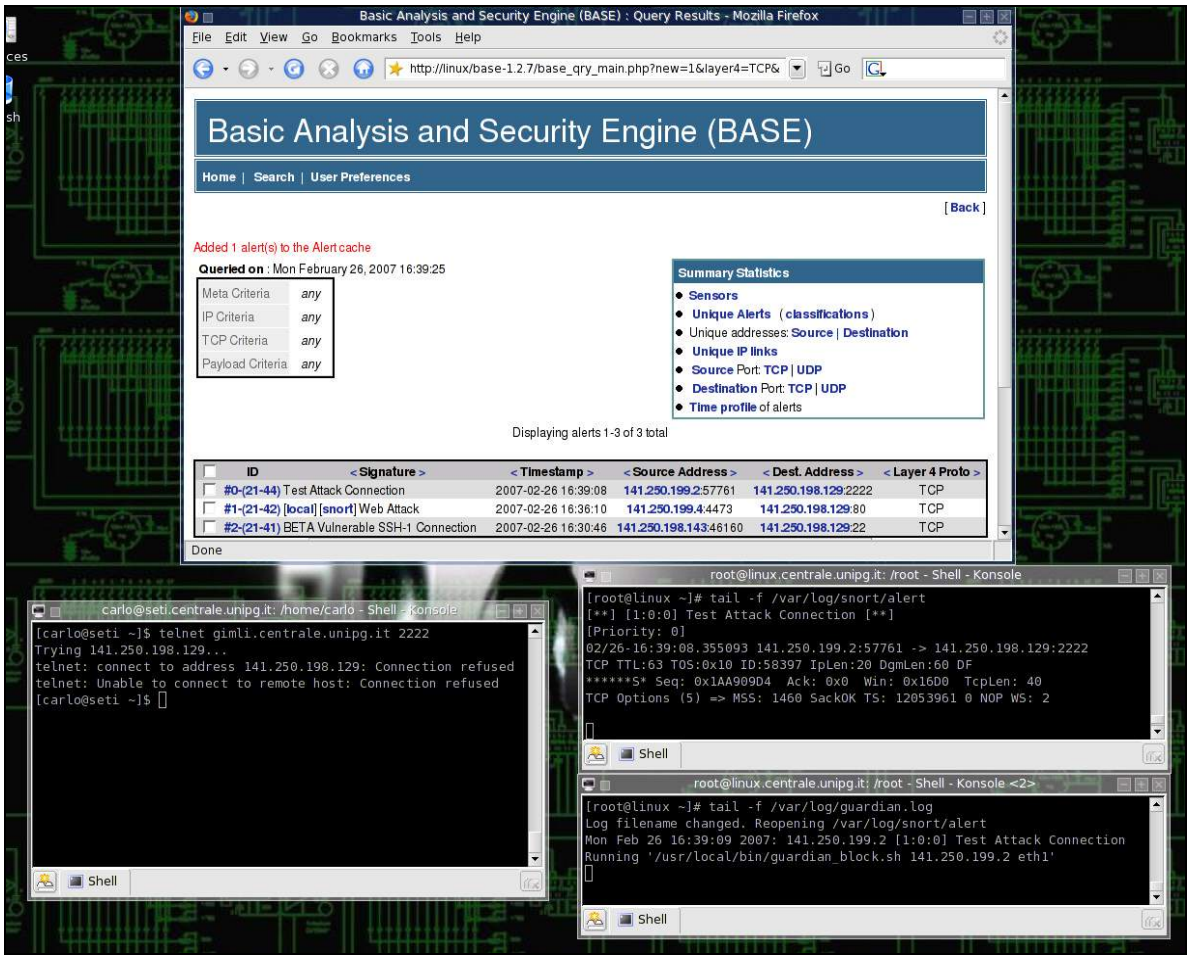

Fig. 13 – Un esempio di funzionamento.

ISSN: 2279–8773

In fig.13 è possibile notare, attraverso la shell, ed in particolare mediante il comando `telnet()`, un tentativo di connessione verso la porta `2222` dell'host `gimli.centrale.unipg.it` appartenente alle subnet monitorata, l'identificazione dell'attacco riportato nel file di log di Snort `$VARLOG/snort/alert`, l'avvio della script di blocking `$LOCALBIN/guardian_block.sh` registrato nel file `$VARLOG/guardian.log`, ed infine, sulla parte superiore della figura, la visualizzazione dell'alert sulla console di BASE.

A tal fine si noti la scritta in rosso '`Added 1 alert(s) to the Alert cache`' che evidenzia la registrazione dell'evento appena verificatosi.

Tale connessione corrisponde infatti ad una delle tre regole personalizzate, specificate nel file `unipg.rules`, definite al fine di verificare il corretto funzionamento di base del sistema.

Per concludere, in fig.14 è invece possibile osservare l'Intrusion Prevention System in produzione che riporta sulla console di BASE un insieme di alert relativi al traffico

ISSN: 2279-8773

di rete preso in esame.

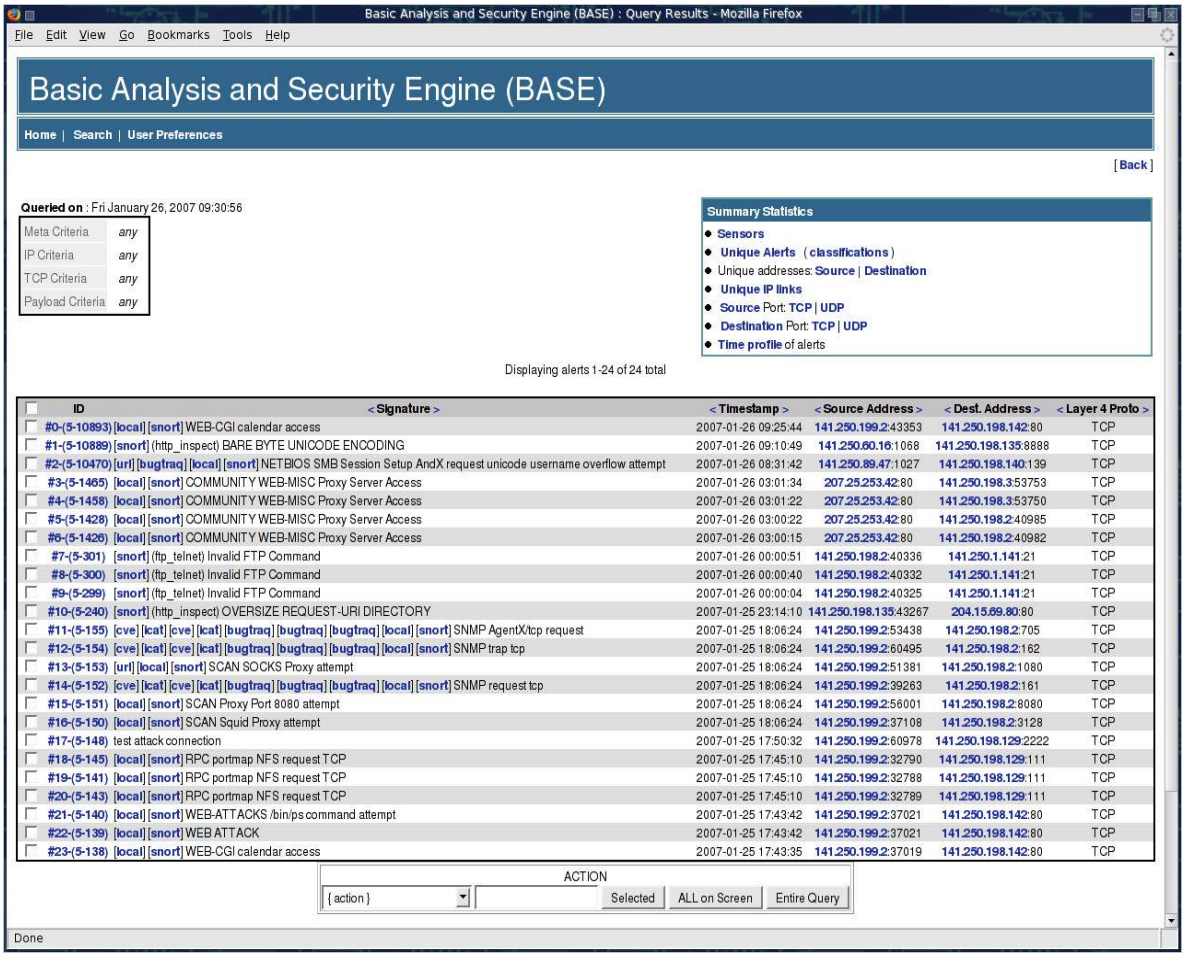

Fig. 14 – L'Intrusion Prevention System (IPS) realizzato.

ISSN: 2279–8773

Conclusioni

Nello svolgimento del presente lavoro di tesi si è tentato di tradurre una molteplicità di concetti teorici espressi in letteratura e riportati nei primi due capitoli in una realizzazione pratica ed originale di un sistema di rilevazione delle intrusioni che realizzi una serie di reazioni dinamiche al fine di classificare il sistema presentato in una contesto più specializzato di prevenzione oltre che di mera e semplice individuazione degli accessi non autorizzati.

Era forte quindi l'esigenza di presentare prima di tutto un approccio efficace ed innovativo.

A tal fine si sono sperimentate le diverse interazioni possibili con un firewall perimetrale, un router di frontiera ed un dispositivo GSM.

Tale sperimentazione, terminata con successo, ha portato alla definizione di un Intrusion Prevention System (IPS) basato interamente su tecnologie e software Open Source, la cui realizzazione ed utilizzo non ha comportato e non

ISSN: 2279-8773

comporta alcun tipo di costo di licensing sotto qualsiasi forma.

Avendo quindi messo in produzione tale sistema si denota, a fianco delle ottime caratteristiche funzionali evidenziate dai diversi software riferiti dal presente lavoro di tesi e presentati nel terzo capitolo, anche una particolare attenzione della comunità di Internet Security rivolta all'aggiornamento dei sistemi di auditing utilizzati, i quali riportano un elevato grado di responsività, traducibile in particolare nel rilascio pressoché immediato di tutti i necessari aggiornamenti in relazione ad ogni nuova minaccia che si presenta e che si diffonde quotidianamente in rete.

Infine una nota conclusiva, da porre come questione aperta agli sviluppatori, ai tecnici del settore ed a tutti coloro che contribuiscono o che volessero contribuire al progetto presentato, riguarda la possibilità di esplorare e quindi di realizzare un sistema di auto-apprendimento che sia in grado di ridurre sensibilmente i costi derivanti dalla fase di tuning, la cui realizzazione, presentata nel terzo paragrafo

ISSN: 2279-8773

del quarto ed ultimo capitolo del presente lavoro di tesi, si è rivelata di fatto notevolmente complessa oltre che significatamente onerosa sia in termini di tempo che di gestione.

ISSN: 2279-8773

Bibliografia

[1] Department of Defense, 'Trusted Computer System Evaluation Criteria', Dicembre 1985.

[2] V. D.Gligor, 'A Note on the Denial-of-Service Problem', Proceedings of the IEEE Symposium on Security and Privacy, pag. 139 –149, 1983.

[3] E. G. Amoroso, 'Fundamentals of Computer Security Technology', Prentice-Hall PTR, 1994.

[4] D. E. Bell, L. J. LaPadula, 'Secure Computer Systems: Unified Exposition and Multics Interpretation', technical report, Mitre TR-2997, Mitre Corporation, Bedford, Marzo 1976.

ISSN: 2279-8773

[5] D. L. Lough, 'A taxonomy of computer attacks with applications to wireless networks', Tesi di Dottorato, Virginia Polytechnic Institute and State University, Aprile 2001.

[6] J. D. Howard, 'An Analysis Of Security Incidents On The Internet, 1989 – 1995', Tesi di Dottorato, università di Pittsburgh, 7 aprile 1997.
http://www.cert.org/research/JHThesis/Start.html

[7] S. Zanero, 'Un Sistema di Intrusion Detection basato su tecniche di apprendimento non supervisionato', Tesi di Laurea, 2002.

[8] Sun Tzu, 'L'arte della guerra', SuperBUR Classici, RCS Milano, Aprile 1999.

[9] M. Strano, 'Computer crime', edizioni Apogeo, Milano, 2000.

ISSN: 2279-8773

[10] D. A. Wheeler, 'Secure Programming for Linux and Unix HOWTO'
http://www.dwheeler.com/secure-programs

[11] E. "Aleph1" Levy, 'Smashing the stack for fun and profit', Phrack magazine, vol. 7, issue 49, Novembre 1996.

[12] SANS Top-20 Internet Security Attack Targets (Annual Update), 2006.
http://www.sans.org/top20

[13] E. H. Spafford, K. A. Heaphy, D. J. Ferbrache, 'A Computer Virus Primer', in 'Computers Under Attack: Intruders, Worms, and Viruses', Peter J. Denning, ed., pp. 316-355, ACM Press, New York, 1990.

[14] A. Ghirardini, 'Social Engineering: una guida introduttiva', technical white paper #3, Italian Black Hats Association, Settembre 2002.
http://www.blackhats.it/it/papers/social_engineering.pdf

ISSN: 2279-8773

[15] James P. Anderson, 'Computer Security Threat Monitoring and Surveillance', Technical report, James P Anderson Co., Fort Washington, Pennsylvania, Aprile 1980.

[16] Gruppo italiano s0ftpj in PROscan.
http://www.s0ftpj.org/tools/proscan.c

[17] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, 'A Network Security Monitor', Proceedings of the IEEE Symposium on Research in Security and Privacy, pag. 296-304, Maggio 1990.
http://seclab.cs.ucdavis.edu/papers/pdfs/th-gd-90.pdf

[18] T. Bass, 'Intrusion detection systems & multisensor data fusion: Creating cyberspace situational awareness', Communications of the ACM, Aprile 2000.

[19] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, T. Grance, L. T. Heberlein, Che-Lin Ho, K. N. Levitt, B. Mukherjee, D. L. Mansur, K. L. Pon, S. E. Smaha, 'A system

ISSN: 2279-8773

for distributed intrusion detection', COMPCOM Spring '91 Digest of Papers, pag. 170-176, Febbraio/Marzo 1991.

[20] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isaco, E. H. Spafford, D. Zamboni, 'An Architecture for Intrusion Detection Using Autonomous Agents', Proceedings of ACSAC '98, pag. 13-24, 1998.

[21] V. Paxson, 'Bro: A System for Detecting Network Intruders in Real-Time', 7th Annual USENIX Security Symposium, 1998.

[22] http://www.ktwo.ca/c/ADMmutate-0.8.4.tar.gz

[23] T. H. Ptacek, T. N. Newsham, 'Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection', Technical Report, Secure Networks, Gennaio 1998.
http://insecure.org/stf/secnet_ids/secnet_ids.html

ISSN: 2279-8773

[24] Snort - The de facto standard for intrusion detection/prevention.

http://www.snort.org

[25] Oinkmaster script.

http://oinkmaster.sourceforge.net

[26] MySQL - The world's most popular open source database.

http://www.mysql.com

[27] BASE - Basic Analysis and Security Engine (BASE) project.

http://base.secureideas.net

[28] Apache - The Apache Software Foundation.

http://www.apache.org

[29] Guardian - Guardian Active Response for Snort.

http://www.chaotic.org/guardian

ISSN: 2279-8773

[30] Gnokii - Open source tools for your mobile phone.

http://www.gnokii.org

[31] Linux Iptables HOWTO.

http://www.linuxguruz.com/iptables/howto

[32] Netfilter/Iptables project homepage.

http://www.netfilter.org

[33] Iproute2 tool.

http://linux-net.osdl.org/index.php/Iproute2

[34] IPROUTE2 Utility Suite Documentation.

http://www.policyrouting.org/iproute2.doc.html

[35] HOWTO Snort.

http://gentoo-wiki.com/HOWTO_Snort

[36] Libnet Library.

http://libnet.sourceforge.net

ISSN: 2279-8773

[37] Libdnet Library.

http://libdnet.sourceforge.net

[38] Libpcap Library.

http://www.tcpdump.org

[39] Snort Users Manual.

http://www.snort.org/docs/snort_manual/2.6.1/snort_manual.pdf

[40] Sourcefire VRT Certified Rules License Agreement.

http://www.snort.org/about_snort/licenses/vrt_license.html

[41] CVS – Open Source Concurrent Versions System.

http://www.nongnu.org/cvs

[42] Sourcefire Vulnerability Research Team (VRT).

http://www.snort.org/vrt

ISSN: 2279-8773

RFC di approfondimento

[1] RFC 1858 - Security Considerations for IP Fragment Filtering.

http://www.faqs.org/rfcs/rfc1858.html

[2] RFC 2179 - Network Security For Trade Shows.

http://www.faqs.org/rfcs/rfc2179.html

[3] RFC 4301 - Security Architecture for the Internet Protocol.

http://www.faqs.org/rfcs/rfc4301.html

[4] RFC 760 - DoD standard Internet Protocol (IP).

http://www.faqs.org/rfcs/rfc760.html

[5] RFC 761 - DoD standard Transmission Control Protocol (TCP).

http://www.faqs.org/rfcs/rfc761.html

ISSN: 2279-8773

Appendice A – Configurazione dei sorgenti di Snort

```
[carlo@linux snort-2.6.0.2]# ./configure --prefix=/usr/local
--includedir=/usr/include/mysql --enable-dynamicplugin --
enable-perfprofiling --enable-timestats --enable-flexresp2 -
-enable-react --enable-linux-smp-stats --with-snmp --with-
mysql --with-openssl |tee configure.log


checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of
Makefiles... no
checking for style of include used by make... GNU
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ANSI C... none needed
checking dependency style of gcc... gcc3
checking for ranlib... ranlib
checking for gcc... (cached) gcc
checking whether we are using the GNU C compiler... (cached)
yes
checking whether gcc accepts -g... (cached) yes
checking for gcc option to accept ANSI C... (cached) none
needed
checking dependency style of gcc... (cached) gcc3
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking for a sed that does not truncate output... /bin/sed
checking for egrep... grep -E
checking for ld used by gcc... /usr/bin/ld
checking if the linker (/usr/bin/ld) is GNU ld... yes
checking for /usr/bin/ld option to reload object files... -r
checking for BSD-compatible nm... /usr/bin/nm -B
checking whether ln -s works... yes
```

ISSN: 2279-8773

```
checking how to recognise dependent libraries... pass_all
checking how to run the C preprocessor... gcc -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking dlfcn.h usability... yes
checking dlfcn.h presence... yes
checking for dlfcn.h... yes
checking for g++... g++
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking dependency style of g++... gcc3
checking how to run the C++ preprocessor... g++ -E
checking for g77... no
checking for f77... no
checking for xlf... no
checking for frt... no
checking for pgf77... no
checking for fort77... no
checking for fl32... no
checking for af77... no
checking for f90... no
checking for xlf90... no
checking for pgf90... no
checking for epcf90... no
checking for f95... no
checking for fort... no
checking for xlf95... no
checking for ifc... no
checking for efc... no
checking for pgf95... no
checking for lf95... no
checking for gfortran... no
checking whether we are using the GNU Fortran 77 compiler...
no
checking whether  accepts -g... no
checking the maximum length of command line arguments...
32768
checking command to parse /usr/bin/nm -B output from gcc
object... ok
checking for objdir... .libs
checking for ar... ar
checking for ranlib... (cached) ranlib
checking for strip... strip
checking if gcc static flag  works... yes
```


ISSN: 2279-8773

```
checking if gcc supports -fno-rtti -fno-exceptions... no
checking for gcc option to produce PIC... -fPIC
checking if gcc PIC flag -fPIC works... yes
checking if gcc supports -c -o file.o... yes
checking whether the gcc linker (/usr/bin/ld) supports
shared libraries... yes
checking whether -lc should be explicitly linked in... no
checking dynamic linker characteristics... GNU/Linux ld.so
checking how to hardcode library paths into programs...
immediate
checking whether stripping libraries is possible... yes
checking if libtool supports shared libraries... yes
checking whether to build shared libraries... yes
checking whether to build static libraries... yes
configure: creating libtool
appending configuration tag "CXX" to libtool
checking for ld used by g++... /usr/bin/ld
checking if the linker (/usr/bin/ld) is GNU ld... yes
checking whether the g++ linker (/usr/bin/ld) supports
shared libraries... yes
checking for g++ option to produce PIC... -fPIC
checking if g++ PIC flag -fPIC works... yes
checking if g++ supports -c -o file.o... yes
checking whether the g++ linker (/usr/bin/ld) supports
shared libraries... yes
checking dynamic linker characteristics... GNU/Linux ld.so
checking how to hardcode library paths into programs...
immediate
checking whether stripping libraries is possible... yes
appending configuration tag "F77" to libtool
checking whether byte ordering is bigendian... no
checking for sparc alignment... no
checking for strings.h... (cached) yes
checking for string.h... (cached) yes
checking for stdlib.h... (cached) yes
checking for unistd.h... (cached) yes
checking sys/sockio.h usability... no
checking sys/sockio.h presence... no
checking for sys/sockio.h... no
checking paths.h usability... yes
checking paths.h presence... yes
checking for paths.h... yes
checking for inet_ntoa in -lnsl... yes
checking for socket in -lsocket... no
checking whether printf must be declared... no
checking whether fprintf must be declared... no
checking whether syslog must be declared... no
checking whether puts must be declared... no
checking whether fputs must be declared... no
checking whether fputc must be declared... no
checking whether fopen must be declared... no
checking whether fclose must be declared... no
```


ISSN: 2279-8773

```
checking whether fwrite must be declared... no
checking whether fflush must be declared... no
checking whether getopt must be declared... no
checking whether bzero must be declared... no
checking whether bcopy must be declared... no
checking whether memset must be declared... no
checking whether strtol must be declared... no
checking whether strcasecmp must be declared... no
checking whether strncasecmp must be declared... no
checking whether strerror must be declared... no
checking whether perror must be declared... no
checking whether socket must be declared... no
checking whether sendto must be declared... no
checking whether vsnprintf must be declared... no
checking whether snprintf must be declared... no
checking whether strtoul must be declared... no
checking for snprintf... yes
checking for strlcpy... no
checking for strlcat... no
checking for strerror... yes
checking for __FUNCTION__... yes
checking for floor in -lm... yes
checking for pcap_datalink in -lpcap... yes
checking pcre.h usability... yes
checking pcre.h presence... yes
checking for pcre.h... yes
checking for pcre_compile in -lpcre... yes
checking for mysql... yes
checking for compress in -lz... yes
checking for dlsym in -ldl... yes
checking dnet.h usability... yes
checking dnet.h presence... yes
checking for dnet.h... yes
checking for eth_set in -ldnet... yes
checking libnet.h usability... yes
checking libnet.h presence... yes
checking for libnet.h... yes
checking for libnet version 1.0.2a... yes
checking for libnet_build_ip in -lnet... yes
checking for u_int8_t... yes
checking for u_int16_t... yes
checking for u_int32_t... yes
checking for a BSD-compatible install... /usr/bin/install -c
configure: creating ./config.status
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating src/sfutil/Makefile
config.status: creating src/detection-plugins/Makefile
config.status: creating src/dynamic-plugins/Makefile
config.status:             creating             src/dynamic-
plugins/sf_engine/Makefile
```

```
config.status:             creating             src/dynamic-
plugins/sf_engine/examples/Makefile
config.status:             creating             src/dynamic-
plugins/sf_preproc_example/Makefile
config.status: creating src/dynamic-preprocessors/Makefile
config.status:             creating             src/dynamic-
preprocessors/ftptelnet/Makefile
config.status:             creating             src/dynamic-
preprocessors/smtp/Makefile
config.status:             creating             src/dynamic-
preprocessors/dns/Makefile
config.status: creating src/dynamic-examples/Makefile
config.status:     creating     src/dynamic-examples/dynamic-
preprocessor/Makefile
config.status:     creating     src/dynamic-examples/dynamic-
rule/Makefile
config.status: creating src/output-plugins/Makefile
config.status: creating src/preprocessors/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/include/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/utils/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/anomaly_detection/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/client/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/event_output/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/mode_inspection/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/normalization/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/server/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/session_inspection/Makefile
config.status:                                          creating
src/preprocessors/HttpInspect/user_interface/Makefile
config.status: creating src/preprocessors/flow/Makefile
config.status:      creating      src/preprocessors/flow/int-
snort/Makefile
config.status:                                          creating
src/preprocessors/flow/portscan/Makefile
config.status: creating src/parser/Makefile
config.status: creating doc/Makefile
config.status: creating contrib/Makefile
config.status: creating schemas/Makefile
config.status: creating rpm/Makefile
config.status: creating m4/Makefile
config.status: creating etc/Makefile
```


ISSN: 2279-8773

```
config.status: creating templates/Makefile
config.status: creating src/win32/Makefile
config.status: creating config.h
config.status: config.h is unchanged
config.status: executing depfiles commands
```


ISSN: 2279-8773

Appendice B – Output e statistiche del sistema alla partenza ed all'arresto dell'IPS

```
[root@linux src]# tail –f /var/log/messages


Running in IDS mode
        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file /etc/snort/snort.conf
Var 'eth1_ADDRESS' defined, value len = 20 chars, value =
[141.250.198.143/32]
Var 'HOME_NET' defined, value len = 35 chars, value =
[141.250.198.0/24,141.250.199.0/26]
Var 'EXTERNAL_NET' defined, value len = 3 chars, value = any
Var 'DNS_SERVERS' defined, value len = 35 chars, value =
[141.250.198.0/24,141.250.199.0/26]
Var 'SMTP_SERVERS' defined, value len = 35 chars, value =
[141.250.198.0/24,141.250.199.0/26]
Var 'HTTP_SERVERS' defined, value len = 35 chars, value =
[141.250.198.0/24,141.250.199.0/26]
Var 'SQL_SERVERS' defined, value len = 35 chars, value =
[141.250.198.0/24,141.250.199.0/26]
Var 'TELNET_SERVERS' defined, value len = 35 chars, value =
[141.250.198.0/24,141.250.199.0/26]
Var 'SNMP_SERVERS' defined, value len = 35 chars, value =
[141.250.198.0/24,141.250.199.0/26]
Var 'HTTP_PORTS' defined, value len = 2 chars, value = 80
Var 'SHELLCODE_PORTS' defined, value len = 3 chars, value =
!80
Var 'ORACLE_PORTS' defined, value len = 4 chars, value =
1521
Var 'AIM_SERVERS' defined, value len = 185 chars, value =
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,6
4.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,2
05.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.
0/24]
```

ISSN: 2279-8773

```
Var 'RULE_PATH' defined, value len = 16 chars, value =
/etc/snort/rules
,-----------[Flow Config]----------------------
| Stats Interval:  0
| Hash Method:     2
| Memcap:          10485760
| Rows  :          4099
| Overhead Bytes:  16400(%0.16)
`-----------------------------------------------
Frag3 global config:
    Max frags: 65536
    Fragment memory cap: 4194304 bytes
Frag3 engine config:
    Target-based policy: FIRST
    Fragment timeout: 60 seconds
    Fragment min_ttl:   1
    Fragment ttl_limit: 5
    Fragment Problems: 1
    Bound Addresses: 0.0.0.0/0.0.0.0
Stream4 config:
    Stateful inspection: ACTIVE
    Session statistics: INACTIVE
    Session timeout: 30 seconds
    Session memory cap: 8388608 bytes
    Session count max: 8192 sessions
    Session cleanup count: 5
    State alerts: INACTIVE
    Evasion alerts: INACTIVE
    Scan alerts: INACTIVE
    Log Flushed Streams: INACTIVE
    MinTTL: 1
    TTL Limit: 5
    Async Link: 0
    State Protection: 0
    Self preservation threshold: 50
    Self preservation period: 90
    Suspend threshold: 200
    Suspend period: 30
    Enforce TCP State: INACTIVE
    Midstream Drop Alerts: INACTIVE
    Server Data Inspection Limit: -1
WARNING /etc/snort/snort.conf(413) => flush_behavior set in
config file, using old static flushpoints (0)
Stream4_reassemble config:
    Server reassembly: INACTIVE
    Client reassembly: ACTIVE
    Reassembler alerts: ACTIVE
    Zero out flushed packets: INACTIVE
    Flush stream on alert: INACTIVE
    flush_data_diff_size: 500
    Reassembler Packet Preferance : Favor Old
    Packet Sequence Overlap Limit: -1
```


ISSN: 2279-8773

```
    Flush behavior: Small (<255 bytes)
    Ports: 21 23 25 42 53 80 110 111 135 136 137 139 143 445
513 1433 1521 3306
    Emergency Ports: 21 23 25 42 53 80 110 111 135 136 13 7
139 143 445 513 1433 1521 3306
HttpInspect Config:
    GLOBAL CONFIG
      Max Pipeline Requests:    0
      Inspection Type:          STATELESS
      Detect Proxy Usage:       NO
      IIS Unicode Map Filename: /etc/snort/unicode.map
      IIS Unicode Map Codepage: 1252
    DEFAULT SERVER CONFIG:
      Ports: 80 8888
      Flow Depth: 300
      Max Chunk Length: 500000
      Inspect Pipeline Requests: YES
      URI Discovery Strict Mode: NO
      Allow Proxy Usage: NO
      Disable Alerting: NO
      Oversize Dir Length: 500
      Only inspect URI: NO
      Ascii: YES alert: NO
      Double Decoding: YES alert: YES
      %U Encoding: YES alert: YES
      Bare Byte: YES alert: YES
      Base36: OFF
      UTF 8: OFF
      IIS Unicode: YES alert: YES
      Multiple Slash: YES alert: NO
      IIS Backslash: YES alert: NO
      Directory Traversal: YES alert: NO
      Web Root Traversal: YES alert: YES
      Apache WhiteSpace: YES alert: NO
      IIS Delimiter: YES alert: NO
      IIS Unicode Map: GLOBAL IIS UNICODE MAP CONFIG
      Non-RFC Compliant Characters: NONE
      Whitespace Characters: 0x09 0x0b 0x0c 0x0d
rpc_decode arguments:
    Ports to decode RPC on: 111 32771
    alert_fragments: INACTIVE
    alert_large_fragments: ACTIVE
    alert_incomplete: ACTIVE
    alert_multiple_requests: ACTIVE
Portscan Detection Config:
    Detect Protocols:  TCP UDP ICMP IP
    Detect  Scan  Type:   portscan  portsweep  decoy_portscan
distributed_portscan
    Sensitivity Level: Low
    Memcap (in bytes): 10000000
    Number of Nodes:   36900
    Log           /var/log/snort/unipg-portscan.log
```


ISSN: 2279-8773

```
    Ignore Scanned IP List:
        141.250.198.131 / 255.255.255.255
Tagged Packet Limit: 256
+----------------------[thresholding-config]---------------
-------------------
| memory-cap : 1048576 bytes
+----------------------[thresholding-global]---------------
-------------------
| none
+----------------------[thresholding-local]----------------
-------------------
|   gen-id=1               sig-id=100000208     type=Threshold
tracking=src count=50  seconds=60
[...] 285 list rules loaded [...]

+----------------------[suppression]-----------------------
-------------------
| none
-----------------------------------------------------------
-------------------
Rule  application  order:  ->activation->dynamic->pass->drop-
>alert->log
Log directory = /var/log/snort
Loading dynamic engine
/usr/local/lib/snort_dynamicengine/libsf_engine.so...
done
Loading all dynamic preprocessor libs from
/usr/local/lib/snort_dynamicpreprocessor/...
  Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_ftptelnet_pr
eproc.so...
done
  Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_smtp_preproc
.so...
done
  Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_dns_preproc.
so...
done
  Finished Loading all dynamic preprocessor libs from
/usr/local/lib/snort_dynamicpreprocessor/
FTPTelnet Config:
    GLOBAL CONFIG
      Inspection Type: stateful
      Check for Encrypted Traffic: YES alert: YES
      Continue to check encrypted data: NO
    TELNET CONFIG:
      Ports: 23
      Are You There Threshold: 200
      Normalize: YES
    FTP CONFIG:
```


ISSN: 2279-8773

```
      FTP Server: default
        Ports: 21
        Check for Telnet Cmds: YES alert: YES
        Identify open data channels: YES
      FTP Client: default
        Check for Bounce Attacks: YES alert: YES
        Check for Telnet Cmds: YES alert: YES
        Max Response Length: 256
SMTP Config:
      Ports: 25
      Inspection Type:            STATEFUL
      Normalize Spaces:           YES
      Ignore Data:                NO
      Ignore TLS Data:            NO
      Ignore Alerts:              NO
      Max Command Length:         0
      Max Header Line Length:     0
      Max Response Line Length:   0
      X-Link2State Alert:         YES
      Drop on X-Link2State Alert: NO
DNS config:
    DNS Client rdata txt Overflow Alert: ACTIVE
    Obsolete DNS RR Types Alert: INACTIVE
    Experimental DNS RR Types Alert: INACTIVE
    Ports: 53
Warning: flowbits key 'community_uri.size.1050' is set but
not ever checked.
Warning: flowbits key 'ms_sql_seen_dns' is checked but not
ever set.
Warning: flowbits key 'smb.tree.create.llsrpc' is set but
not ever checked.
Initializing daemon mode
Var 'eth1_ADDRESS' redefined
PID path stat checked out ok, PID path set to /var/run/
Writing PID "9602" to file "/var/run//snort_eth1.pid"
Daemon parent exiting
Daemon initialized, signaled parent pid: 9601
Snort initialization completed successfully (pid=9602)
Not Using PCAP_FRAMES
linux kernel: eth1: Setting promiscuous mode.
linux kernel: device eth1 entered promiscuous mode
```


ISSN: 2279-8773

[...WORKING..] -> now press CTRL+C during foreground mode or stop daemon otherwise

```
*** Caught Term-Signal
Frag3 statistics:
        Total Fragments: 0
      Frags Reassembled: 0
               Discards: 0
          Memory Faults: 0
               Timeouts: 0
               Overlaps: 0
              Anomalies: 0
                 Alerts: 0
     FragTrackers Added: 0
    FragTrackers Dumped: 0
FragTrackers Auto Freed: 0
    Frag Nodes Inserted: 0
     Frag Nodes Deleted: 0
===================================================================
==================
Final Flow Statistics
,----[ FLOWCACHE STATS ]----------
Memcap: 10485760 Overhead Bytes 16400 used(%1.713257)/blocks
(179648/913) Overhead blocks: 1 Could Hold: (58579)
IPV4 count: 912 frees: 0 low_time: 1171881462, high_time:
1171882154, diff: 0h:11:32s
    finds: 384695 reversed: 176197(%45.801739)
find_success: 383783 find_fail: 912 percent_success:
(%99.762929) new_flows: 912
 Protocol: 1 (%0.000260)    finds: 1    reversed:
0(%0.000000)    find_success: 0    find_fail: 1
percent_success: (%0.000000)    new_flows: 1
 Protocol: 6 (%99.831555)    finds: 384047    reversed:
175983(%45.823298)    find_success: 383225    find_fail: 822
percent_success: (%99.785964)    new_flows: 822
 Protocol: 17 (%0.168185)    finds: 647    reversed:
214(%33.075734)    find_success: 558    find_fail: 89
percent_success: (%86.244204)    new_flows: 89
Snort ran for 0 Days 0 Hours 12 Minutes 18 Seconds
Packet analysis time averages:
Snort Analyzed 57654 Packets Per Minute
Snort Analyzed 937 Packets Per Second
Snort received 691854 packets
    Analyzed: 340355(49.195%)
    Dropped: 11133(1.609%)
    Outstanding: 340366(49.196%)
===================================================================
==================
Breakdown by protocol:
    TCP: 338675     (99.506%)
    UDP: 647        (0.190%)
```


ISSN: 2279-8773

```
   ICMP: 1          (0.000%)
    ARP: 663        (0.195%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
ETHLOOP: 0          (0.000%)
    IPX: 0          (0.000%)
   FRAG: 0          (0.000%)
  OTHER: 369        (0.108%)
DISCARD: 0          (0.000%)
===================================================================
===================
Action Stats:
ALERTS: 3
LOGGED: 3
PASSED: 6916
===================================================================
===================
TCP Stream Reassembly Stats:
    TCP Packets Used: 338675     (99.506%)
    Stream Trackers: 1032
    Stream flushes: 45372
    Segments used: 132208
    Stream4 Memory Faults: 0
===================================================================
===================
Snort exiting
linux kernel: device eth1 left promiscuous mode
```


ISSN: 2279-8773

Appendice C – Identificazione e test del dispositivo GSM

```
[root@linux src]# gnokii --identify

GNOKII Version 0.6.4
LOG: debug mask is 0x1
phone instance config:
model: 3310
port_device: /dev/ttyS0
connection_type: 0
init_length: 0
serial_baudrate: 19200
serial_write_usleep: -1
hardware_handshake: 0
require_dcd: 0
smsc_timeout: 100
connect_script:
disconnect_script:
rfcomm_cn: 1
sm_retry: off
Serial device: opening device /dev/ttyS0
Serial device: setting speed to 115200
Serial device: setting RTS to low and DTR to high
Message sent: 0x40 / 0x0004
00 01 64 01                                       |   d
[Received Ack of type 40, seq:  0]
[Sending Ack of type 40, seq: 1]
Message received: 0x40 / 0x000c
01 01 64 03 01 4f 0d 01 01 01 1b 58               |   d  O
X
Received message type 40
Message: Extended commands enabled.
Message sent: 0x40 / 0x0003
00 01 66                                          |   f
[Received Ack of type 40, seq:  1]
[Sending Ack of type 40, seq: 2]
Message received: 0x40 / 0x0014
01  01  66  01  33  35  32  39  33  38  30  30  31  33  36  33 |    f
352938001363
34 34 34 00                                       | 444
Received message type 40
```

```
IMEI: 352938001363444
Message sent: 0x40 / 0x0004
00 01 64 01                                       |   d
[Received Ack of type 40, seq:  2]
[Sending Ack of type 40, seq: 3]
Message received: 0x40 / 0x000c
01 01 64 03 01 4f 0d 01 01 01 1b 58               |   d  O
X
Received message type 40
Message: Extended commands enabled.
Message sent: 0x40 / 0x0004
00 01 c8 01                                       |
[Received Ack of type 40, seq:  3]
[Sending Ack of type 40, seq: 4]
Message received: 0x40 / 0x0025
01 01 c8 01 00 56 20 30 36 2e 30 37 0a 31 37 2d | V
06.07 17-
30 36 2d 30 33 0a 4e 48 4d 2d 35 0a 28 63 29 20 | 06-03 NHM-
5 (c)
4e 4d 50 2e 00                                    | NMP.
Received message type 40
Received SW 06.07
Received model NHM-5
Message sent: 0x40 / 0x0004
00 01 64 01                                       |   d
[Received Ack of type 40, seq:  4]
[Sending Ack of type 40, seq: 5]
Message received: 0x40 / 0x000c
01 01 64 03 01 4f 0d 01 01 01 1b 58               |   d  O
X
Received message type 40
Message: Extended commands enabled.
Message sent: 0x40 / 0x0004
00 01 c8 05                                       |
[Received Ack of type 40, seq:  5]
[Sending Ack of type 40, seq: 6]
Message received: 0x40 / 0x000a
01 01 c8 05 00 31 30 35 34 00                     |      1054
Received message type 40
Received SW 06.07, HW 1054
comparing "NHM-5" and ""
comparing "NHM-5" and "?????"
comparing "NHM-5" and "?????"
comparing "NHM-5" and "TFF-3"
comparing "NHM-5" and "RX-2"
comparing "NHM-5" and "TME-2"
comparing "NHM-5" and "TME-1"
comparing "NHM-5" and "TME-3"
comparing "NHM-5" and "THX-9L"
comparing "NHM-5" and "THF-9"
comparing "NHM-5" and "NHX-8"
comparing "NHM-5" and "THF-11"
```


ISSN: 2279-8773

```
comparing "NHM-5" and "THF-10"
comparing "NHM-5" and "THF-13"
comparing "NHM-5" and "THF-12"
comparing "NHM-5" and "TFE-4R"
comparing "NHM-5" and "NHE-2"
comparing "NHM-5" and "RH-18"
comparing "NHM-5" and "RH-38"
comparing "NHM-5" and "RH-36"
comparing "NHM-5" and "NKC-1"
comparing "NHM-5" and "NKW-1"
comparing "NHM-5" and "NKW-1C"
comparing "NHM-5" and "NHE-5"
comparing "NHM-5" and "NHE-5NX"
comparing "NHM-5" and "NHE-5"
comparing "NHM-5" and "NHE-5NA"
comparing "NHM-5" and "NHE-5NX"
comparing "NHM-5" and "NHE-5SA"
comparing "NHM-5" and "NHE-3"
comparing "NHM-5" and "NAM-2"
comparing "NHM-5" and "NHE-1"
comparing "NHM-5" and "NHE-4"
comparing "NHM-5" and "NHE-4"
comparing "NHM-5" and "NHK-1XA"
comparing "NHM-5" and "NHK-1"
comparing "NHM-5" and "NHK-4"
comparing "NHM-5" and "NHC-4NE/HE"
comparing "NHM-5" and "NHC-4NE/HE"
comparing "NHM-5" and "NHP-4"
comparing "NHM-5" and "NHD-4X"
comparing "NHM-5" and "NHB-3NB"
comparing "NHM-5" and "RH-40"
comparing "NHM-5" and "RH-42"
comparing "NHM-5" and "RH-39"
comparing "NHM-5" and "RH-41"
comparing "NHM-5" and "RH-3P"
comparing "NHM-5" and "RH-3DNG"
comparing "NHM-5" and "RH-17"
comparing "NHM-5" and "RH-3"
comparing "NHM-5" and "RM-4"
comparing "NHM-5" and "RM-5"
comparing "NHM-5" and "RH-19"
comparing "NHM-5" and "RH-50"
comparing "NHM-5" and "RH-48"
comparing "NHM-5" and "RH-6"
comparing "NHM-5" and "NHE-8"
comparing "NHM-5" and "0310"
comparing "NHM-5" and "RH-30"
comparing "NHM-5" and "RH-31"
comparing "NHM-5" and "NSE-8"
comparing "NHM-5" and "NSE-9"
comparing "NHM-5" and "NSD-1A"
comparing "NHM-5" and "NEM-1"
```


ISSN: 2279-8773

```
comparing "NHM-5" and "NEM-2"
Found model "NHM-5"
IMEI         : 352938001363444
Manufacturer : Nokia
Model        : NHM-5
Revision     : SW 06.07, HW 1054
Serial device: closing device


[root@linux src]# echo "The First IPS message via GSM" |
gnokii --sendsms +393465313707 --smsc +393359609600


GNOKII Version 0.6.4
LOG: debug mask is 0x1
phone instance config:
model: 3310
port_device: /dev/ttyS0
connection_type: 0
init_length: 0
serial_baudrate: 19200
serial_write_usleep: -1
hardware_handshake: 0
require_dcd: 0
smsc_timeout: 100
connect_script:
disconnect_script:
rfcomm_cn: 1
sm_retry: off
Serial device: opening device /dev/ttyS0
Serial device: setting speed to 115200
Serial device: setting RTS to low and DTR to high
Message sent: 0x40 / 0x0004
00 01 64 01                                       |   d
[Received Ack of type 40, seq:  0]
[Sending Ack of type 40, seq: 7]
Message received: 0x40 / 0x000c
01 01 64 03 01 4f 0d 01 01 01 1b 58               |   d  O
X
Received message type 40
Message: Extended commands enabled.
Message sent: 0x40 / 0x0003
00 01 66                                          |   f
[Received Ack of type 40, seq:  1]
[Sending Ack of type 40, seq: 0]
Message received: 0x40 / 0x0014
01  01  66  01  33  35  32  39  33  38  30  30  31  33  36  33 |    f
352938001363
34 34 34 00                                       | 444
Received message type 40
IMEI: 352938001363444
```


ISSN: 2279–8773

```
Message sent: 0x40 / 0x0004
00 01 64 01                                       |   d
[Received Ack of type 40, seq:  2]
[Sending Ack of type 40, seq: 1]
Message received: 0x40 / 0x000c
01 01 64 03 01 4f 0d 01 01 01 1b 58               |   d  O
X
Received message type 40
Message: Extended commands enabled.
Message sent: 0x40 / 0x0004
00 01 c8 01                                       |
[Received Ack of type 40, seq:  3]
[Sending Ack of type 40, seq: 2]
Message received: 0x40 / 0x0025
01 01 c8 01 00 56 20 30 36 2e 30 37 0a 31 37 2d |       V
06.07 17-
30 36 2d 30 33 0a 4e 48 4d 2d 35 0a 28 63 29 20 | 06-03 NHM-
5 (c)
4e 4d 50 2e 00                                    | NMP.
Received message type 40
Received SW 06.07
Received model NHM-5
Message sent: 0x40 / 0x0004
00 01 64 01                                       |   d
[Received Ack of type 40, seq:  4]
[Sending Ack of type 40, seq: 3]
Message received: 0x40 / 0x000c
01 01 64 03 01 4f 0d 01 01 01 1b 58               |   d  O
X
Received message type 40
Message: Extended commands enabled.
Message sent: 0x40 / 0x0004
00 01 c8 05                                       |
[Received Ack of type 40, seq:  5]
[Sending Ack of type 40, seq: 4]
Message received: 0x40 / 0x000a
01 01 c8 05 00 31 30 35 34 00                     |      1054
Received message type 40
Received SW 06.07, HW 1054
Found model "NHM-5"
General Data Coding
dcs: 0x0
Length: 0x1d
user_data_length: 0x1a
ValidityIndicator: 2
user_data:
547419644CCBE77450123A05B5CBF379F85C06D9D361D071DA04
Sending
Message sent: 0x0a / 0x0004
00 01 00 70                                       |    p
[Received Ack of type 0a, seq:  6]
[Sending Ack of type 0a, seq: 5]
```


ISSN: 2279-8773

```
Message received: 0x0a / 0x0013
01 08 00 71 01 00 01 0b 01 02 1d 20 b0 e9 22 f2 |     q
"
10 00 00                                        |
Received message type 0a
Message sent: 0x02 / 0x0044
00 01 00 01 02 00 07 91 93 33 95 06 69 00 00 00 |          3
i
00  00  11  00  00  00  1d  0c  91  93  33  59  20  20  71  00  |
3Y  q
00  00  00  a9  00  00  00  00  00  00  54  74  19  64  4c  cb  |
Tt dL
e7 74 50 12 3a 05 b5 cb f3 79 f8 5c 06 d9 d3 61 |  tP :    y
\   a
d0 71 da 04                                     |  q
[Received Ack of type 02, seq:  7]
[Sending Ack of type 02, seq: 6]
Message received: 0x02 / 0x0007
01 08 00 02 64 4c 00                            |     dL
Received message type 02
Send succeeded!
Serial device: closing device
```

Fig. 15 – Il primo messaggio inviato dal sistema IPS.

ISSN: 2279-8773

Appendice D – Tuning:

file `excludes-dpp.conf` e `pass.rules`

```
[root@linux src]# cat /etc/snort/excludes-dpp.conf

not (src host 141.250.198.135 and src portrange 1024-65535
and dst port 80) and
not (dst host 141.250.198.135 and src portrange 1024-65535
and dst port 8888) and
not (src host 141.250.198.2 and src portrange 1024-65535 and
dst host 141.250.1.141 and dst port 21) and
not (src host 141.250.199.2 and src portrange 1024-65535 and
dst host 141.250.198.142 and dst port 80) and
not (src host 141.250.1.141 and src port 21 and dst host
141.250.198.2 and dst portrange 1024-65535) and
not (src net 141.250.199.0/26 and src portrange 1024-65535
and dst host 141.250.198.138 and dst port 80)
```

```
[root@linux src]# cat /etc/snort/pass.rules

# Admin Rules
# seti.centrale.unipg.it - ALL PRIVILEGES on HOME_NET unless
2222 destination port for tests
pass tcp 141.250.84.212 any -> $HOME_NET !2222

# icmp-info.rules
# teseo.unipg.it
pass icmp 141.250.1.7 any -> $HOME_NET any (msg:"ICMP
Destination Unreachable Port Unreachable"; icode:3; itype:3;
classtype:misc-activity; sid:402; rev:7;)
pass icmp 141.250.0.0/16 any -> $HOME_NET any (msg:"ICMP
Echo Reply"; icode:0; itype:0; classtype:misc-activity;
sid:408; rev:5;)
# aragorn.centrale.unipg.it
pass icmp 141.250.198.135 any -> 141.250.0.0/16 any
(msg:"ICMP Destination Unreachable Communication with
Destination Host is Administratively Prohibited"; icode:10;
itype:3; classtype:misc-activity; sid:486; rev:4;)
# custom
pass icmp 141.250.0.0/16 any -> $HOME_NET any (msg:"ICMP
PING"; icode:0; itype:8; classtype:misc-activity; sid:384;
rev:5;)
```

ISSN: 2279–8773

```
pass icmp $HOME_NET any -> $HOME_NET any (msg:"ICMP
L3retriever Ping"; icode:0; itype:8;
content:"ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI"; depth:32;
reference:arachnids,311; classtype:attempted-recon; sid:466;
rev:4;)

# misc.rules
# gandalf.centrale.unipg.it
pass tcp 141.250.0.0/16 any -> 141.250.198.128 3389
(msg:"MISC MS Terminal server request";
flow:to_server,established; content:"|03 00 00|"; depth:3;
content:"|E0 00 00 00 00 00|"; depth:6; offset:5;
reference:bugtraq,3099; reference:cve,2001-0540;
reference:url,www.microsoft.com/technet/security/bulletin/MS
01-040.mspx; classtype:protocol-command-decode; sid:1448;
rev:12;)
pass tcp 141.250.0.0/16 any -> 141.250.198.128 3389
(msg:"MISC MS Terminal Server no encryption session
initiation attempt"; flow:to_server,established;
content:"|03 00 01|"; depth:3; content:"|00|"; depth:1;
offset:288;
reference:url,www.microsoft.com/technet/security/bulletin/MS
01-052.mspx; classtype:attempted-dos; sid:2418; rev:4;)
# arwen.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.131 3389 (msg:"MISC MS
Terminal server request"; flow:to_server,established;
content:"|03 00 00|"; depth:3; content:"|E0 00 00 00 00
00|"; depth:6; offset:5; reference:bugtraq,3099;
reference:cve,2001-0540;
reference:url,www.microsoft.com/technet/security/bulletin/MS
01-040.mspx; classtype:protocol-command-decode; sid:1448;
rev:12;)
# frodo.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.133 3389 (msg:"MISC MS
Terminal server request"; flow:to_server,established;
content:"|03 00 00|"; depth:3; content:"|E0 00 00 00 00
00|"; depth:6; offset:5; reference:bugtraq,3099;
reference:cve,2001-0540;
reference:url,www.microsoft.com/technet/security/bulletin/MS
01-040.mspx; classtype:protocol-command-decode; sid:1448;
rev:12;)
pass tcp $HOME_NET any -> 141.250.198.133 3389 (msg:"MISC MS
Terminal Server no encryption session initiation attempt";
flow:to_server,established; content:"|03 00 01|"; depth:3;
content:"|00|"; depth:1; offset:288;
reference:url,www.microsoft.com/technet/security/bulletin/MS
01-052.mspx; classtype:attempted-dos; sid:2418; rev:4;)
# sam.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.134 3389 (msg:"MISC MS
Terminal server request"; flow:to_server,established;
content:"|03 00 00|"; depth:3; content:"|E0 00 00 00 00
00|"; depth:6; offset:5; reference:bugtraq,3099;
```


ISSN: 2279-8773

```
reference:cve,2001-0540;
reference:url,www.microsoft.com/technet/security/bulletin/MS
01-040.mspx;  classtype:protocol-command-decode;  sid:1448;
rev:12;)
pass tcp $HOME_NET any -> 141.250.198.134 3389 (msg:"MISC MS
Terminal Server no encryption session initiation attempt";
flow:to_server,established; content:"|03 00 01|"; depth:3;
content:"|00|";            depth:1;            offset:288;
reference:url,www.microsoft.com/technet/security/bulletin/MS
01-052.mspx; classtype:attempted-dos; sid:2418; rev:4;)
# www.presenze.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.138 3389 (msg:"MISC MS
Terminal  server  request";  flow:to_server,established;
content:"|03 00 00|"; depth:3; content:"|E0 00 00 00 00
00|";    depth:6;    offset:5;    reference:bugtraq,3099;
reference:cve,2001-0540;
reference:url,www.microsoft.com/technet/security/bulletin/MS
01-040.mspx;  classtype:protocol-command-decode;  sid:1448;
rev:12;)
pass tcp $HOME_NET any -> 141.250.198.138 3389 (msg:"MISC MS
Terminal Server no encryption session initiation attempt";
flow:to_server,established; content:"|03 00 01|"; depth:3;
content:"|00|";            depth:1;            offset:288;
reference:url,www.microsoft.com/technet/security/bulletin/MS
01-052.mspx; classtype:attempted-dos; sid:2418; rev:4;)
# fs.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.140 3389 (msg:"MISC MS
Terminal  server  request";  flow:to_server,established;
content:"|03 00 00|"; depth:3; content:"|E0 00 00 00 00
00|";    depth:6;    offset:5;    reference:bugtraq,3099;
reference:cve,2001-0540;
reference:url,www.microsoft.com/technet/security/bulletin/MS
01-040.mspx;  classtype:protocol-command-decode;  sid:1448;
rev:12;)

# rpc.rules
# tsm.centrale.unipg.it
pass udp 141.250.198.136 any -> $HOME_NET 111 (msg:"RPC
portmap proxy attempt UDP"; content:"|00 01 86 A0|";
depth:4; offset:12; content:"|00 00 00 05|"; within:4;
distance:4; content:"|00 00 00 00|"; depth:4; offset:4;
classtype:rpc-portmap-decode; sid:1923; rev:6;)

# shellcode.rules
# tsm.centrale.unipg.it
pass ip 141.250.0.0/16 $SHELLCODE_PORTS -> 141.250.198.136
1500 (msg:"SHELLCODE x86 NOOP"; content:"|90 90 90 90 90 90
90  90  90  90  90  90  90  90|";  depth:128;
reference:arachnids,181;         classtype:shellcode-detect;
sid:648; rev:7;)
pass ip 141.250.0.0/16 $SHELLCODE_PORTS -> 141.250.198.136
1500 (msg:"SHELLCODE x86 0x90 unicode NOOP"; content:"|90 00
```


ISSN: 2279-8773

```
90 00 90 00 90 00 90 00|"; classtype:shellcode-detect;
sid:653; rev:9;)
pass ip 141.250.0.0/16 $SHELLCODE_PORTS -> 141.250.198.136
1500 (msg:"SHELLCODE x86 inc ebx NOOP";
content:"CCCCCCCCCCCCCCCCCCCCCCCC"; classtype:shellcode-
detect; sid:1390; rev:5;)
pass ip 141.250.0.0/16 $SHELLCODE_PORTS -> 141.250.198.136
1500 (msg:"SHELLCODE x86 NOOP";
content:"aaaaaaaaaaaaaaaaaaaa"; classtype:shellcode-detect;
sid:1394; rev:5;)
# fs.centrale.unipg.it
pass ip 141.250.0.0/16 $SHELLCODE_PORTS -> 141.250.198.140
445 (msg:"SHELLCODE x86 0x90 unicode NOOP"; content:"|90 00
90 00 90 00 90 00|"; classtype:shellcode-detect;
sid:653; rev:9;)
pass ip 141.250.0.0/16 $SHELLCODE_PORTS -> 141.250.198.140
445 (msg:"SHELLCODE x86 NOOP";
content:"aaaaaaaaaaaaaaaaaaaa"; classtype:shellcode-detect;
sid:1394; rev:5;)
pass ip 141.250.198.140 $SHELLCODE_PORTS -> 141.250.199.15
445 (msg:"SHELLCODE x86 0x90 unicode NOOP"; content:"|90 00
90 00 90 00 90 00|"; classtype:shellcode-detect;
sid:653; rev:9;)
pass ip 141.250.198.140 $SHELLCODE_PORTS -> 141.250.199.15
445 (msg:"SHELLCODE x86 inc ebx NOOP";
content:"CCCCCCCCCCCCCCCCCCCCCCCC"; classtype:shellcode-
detect; sid:1390; rev:5;)
pass ip 141.250.198.140 $SHELLCODE_PORTS -> 141.250.199.15
445 (msg:"SHELLCODE x86 NOOP";
content:"aaaaaaaaaaaaaaaaaaaa"; classtype:shellcode-detect;
sid:1394; rev:5;)
pass ip 141.250.198.140 $SHELLCODE_PORTS -> 141.250.199.15
445 (msg:"SHELLCODE x86 NOOP"; content:"|90 90 90 90 90 90
90 90 90 90 90 90 90 90|"; depth:128;
reference:arachnids,181; classtype:shellcode-detect;
sid:648; rev:7;)
pass ip 141.250.198.140 $SHELLCODE_PORTS -> 141.250.199.15
445 (msg:"SHELLCODE x86 setuid 0"; content:"|B0 17 CD 80|";
reference:arachnids,436; classtype:system-call-detect;
sid:650; rev:8;)
pass ip 141.250.198.140 $SHELLCODE_PORTS -> 141.250.199.15
445 (msg:"SHELLCODE x86 setgid 0"; content:"|B0 B5 CD 80|";
reference:arachnids,284; classtype:system-call-detect;
sid:649; rev:8;)

# info.rules
# custom
pass tcp $HOME_NET $HTTP_PORTS -> $HOME_NET any (msg:"INFO
web bug 0x0 gif attempt"; flow:from_server,established;
content:"Content-type|3A| image/gif"; nocase; content:"GIF";
nocase; distance:0; content:"|01 00 01 00|"; distance:3;
within:4; content:"|2C|"; distance:0; content:"|01 00 01
```


ISSN: 2279-8773

```
00|";    distance:4;    within:4;    classtype:misc-activity;
sid:2925; rev:2;)

# community-web-attacks.rules
# custom
pass tcp 141.250.0.0/16 any -> $HOME_NET 80 (msg:"COMMUNITY
WEB-ATTACKS GFI MailSecurity Management Host Overflow
Attempt Long Accept Parameter"; flow:to_server,established;
content:"Accept"; nocase; pcre:"/^Accept[^\r\n]{200,}/smi";
reference:bugtraq,15081;
reference:url,www.osvdb.org/displayvuln.php?osvdb_id=19926;
classtype:attempted-admin; sid:100000171; rev:3;)

# community-web-misc.rules
# aragorn.centrale.unipg.it
pass tcp 141.250.198.135 any -> $EXTERNAL_NET any
(msg:"COMMUNITY WEB-MISC Proxy Server Access";
flow:established,from_server; content:"Proxy-Connection";
nocase; content:"Via"; nocase; content:"HTTP"; nocase;
content: !"ERR_ACCESS_DENIED"; nocase; logto: "proxy";
classtype:misc-activity; sid:100000132; rev:3;)
# fs.centrale.unipg.it
pass tcp 141.250.198.140 any -> 141.250.0.0/16 any
(msg:"COMMUNITY WEB-MISC Proxy Server Access";
flow:established,from_server; content:"Proxy-Connection";
nocase; content:"Via"; nocase; content:"HTTP"; nocase;
content: !"ERR_ACCESS_DENIED"; nocase; logto: "proxy";
classtype:misc-activity; sid:100000132; rev:3;)

# community-misc.rules
# custom
pass tcp 141.250.0.0/16 any -> 141.250.0.0/16 !139
(msg:"COMMUNITY MISC BAD-SSL tcp detect"; flow:stateless;
content:"|00 0E|"; depth:4; offset:0; classtype:misc-
activity; sid:100000137; rev:1;)
# gandalf.centrale.unipg.it
pass tcp 130.186.0.0/19 any -> 141.250.198.128 !139
(msg:"COMMUNITY MISC BAD-SSL tcp detect"; flow:stateless;
content:"|00 0E|"; depth:4; offset:0; classtype:misc-
activity; sid:100000137; rev:1;)
# aragorn.centrale.unipg.it
pass tcp 130.186.0.0/19 443 -> 141.250.198.135 any
(msg:"COMMUNITY MISC BAD-SSL tcp detect"; flow:stateless;
content:"|00 0E|"; depth:4; offset:0; classtype:misc-
activity; sid:100000137; rev:1;)

# web-misc.rules
# giss-svc1.centrale.unipg.it
pass tcp 141.250.0.0/16 any -> 141.250.198.130 443
(msg:"WEB-MISC SSLv3 invalid Client_Hello attempt";
flow:to_server,established,no_stream;
flowbits:isset,sslv3.server_hello.request;      content:"|16
```


ISSN: 2279-8773

```
03|";  depth:2;  content:"|01|";  depth:1;  offset:5;
reference:cve,2004-0120;  reference:nessus,12204;
reference:url,www.microsoft.com/technet/security/bulletin/MS
04-011.mspx; classtype:attempted-dos; sid:2522; rev:12;)
pass  tcp  141.250.0.0/16  any  ->  141.250.198.130  443
(msg:"WEB-MISC  PCT  Client_Hello  overflow  attempt";
flow:to_server,established;  content:"|01|";  depth:1;
offset:2;  byte_test:2,>,0,6;  byte_test:2,!,0,8;
byte_test:2,!,16,8;  byte_test:2,>,20,10;  content:"|8F|";
depth:1;  offset:11;  byte_test:2,>,32768,0,relative;
reference:bugtraq,10116;  reference:cve,2003-0719;
reference:url,www.microsoft.com/technet/security/bulletin/MS
04-011.mspx; classtype:attempted-admin; sid:2515; rev:9;)
# arwen.centrale.unipg.it
pass tcp 141.250.0.0/16 any -> 141.250.198.131 80 (msg:"WEB-
MISC  Lotus  Notes  .exe  script  source  download  attempt";
flow:to_server,established;  uricontent:".exe";
content:".exe";  content:".";  within:1;
reference:bugtraq,6841;  classtype:web-application-attack;
sid:2067; rev:4;)
# www.presenze.centrale.unipg.it
pass  tcp  141.250.0.0/16  any  ->  141.250.198.138  443
(msg:"WEB-MISC  PCT  Client_Hello  overflow  attempt";
flow:to_server,established;  content:"|01|";  depth:1;
offset:2;  byte_test:2,>,0,6;  byte_test:2,!,0,8;
byte_test:2,!,16,8;  byte_test:2,>,20,10;  content:"|8F|";
depth:1;  offset:11;  byte_test:2,>,32768,0,relative;
reference:bugtraq,10116;  reference:cve,2003-0719;
reference:url,www.microsoft.com/technet/security/bulletin/MS
04-011.mspx; classtype:attempted-admin; sid:2515; rev:9;)

# deleted.rules
# custom
pass  ip  141.250.0.0/16  any  ->  $HOME_NET  any  (msg:"BAD
TRAFFIC Non-Standard IP protocol"; ip_proto:!1; ip_proto:!2;
ip_proto:!47;  ip_proto:!50;  ip_proto:!51;  ip_proto:!6;
ip_proto:!89;  classtype:non-standard-protocol;  sid:1620;
rev:5;)
# dedalo.unipg.it e teseo.unipg.it, dns e ntp
pass ip 141.250.1.6 53 -> $HOME_NET any (msg:"BAD TRAFFIC
Non-Standard  IP  protocol";  ip_proto:!1;  ip_proto:!2;
ip_proto:!47;  ip_proto:!50;  ip_proto:!51;  ip_proto:!6;
ip_proto:!89;  classtype:non-standard-protocol;  sid:1620;
rev:5;)
pass ip 141.250.1.6 123 -> $HOME_NET any (msg:"BAD TRAFFIC
Non-Standard  IP  protocol";  ip_proto:!1;  ip_proto:!2;
ip_proto:!47;  ip_proto:!50;  ip_proto:!51;  ip_proto:!6;
ip_proto:!89;  classtype:non-standard-protocol;  sid:1620;
rev:5;)
pass ip 141.250.1.7 53 -> $HOME_NET any (msg:"BAD TRAFFIC
Non-Standard  IP  protocol";  ip_proto:!1;  ip_proto:!2;
ip_proto:!47;  ip_proto:!50;  ip_proto:!51;  ip_proto:!6;
```


ISSN: 2279-8773

```
ip_proto:!89;   classtype:non-standard-protocol;   sid:1620;
rev:5;)
pass ip 141.250.1.7 123 -> $HOME_NET any (msg:"BAD TRAFFIC
Non-Standard   IP   protocol";   ip_proto:!1;   ip_proto:!2;
ip_proto:!47;   ip_proto:!50;   ip_proto:!51;   ip_proto:!6;
ip_proto:!89;   classtype:non-standard-protocol;   sid:1620;
rev:5;)
# gollum.centrale.unipg.it
alert   tcp   141.250.0.0/16   any   ->   141.250.198.130   443
(msg:"WEB-MISC    SSLv3    invalid    timestamp    attempt";
flow:to_server,established;   content:"|16   03|";   depth:2;
content:"|01|";              depth:1;              offset:5;
byte_test:4,>,2147483647,5,relative;
reference:bugtraq,10115;            reference:cve,2004-0120;
reference:nessus,12204;
reference:url,www.microsoft.com/technet/security/bulletin/MS
04-011.mspx; classtype:attempted-dos; sid:2506; rev:10;)
# arwen.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.131 80 (msg:"WEB-IIS
header      field      buffer      overflow      attempt";
flow:to_server,established;  content:"|3A|";  content:"|0A|";
content:"|00|";  reference:bugtraq,4476;  reference:cve,2002-
0150; classtype:web-application-attack; sid:1768; rev:7;)
# frodo.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.133 139 (msg:"NETBIOS
SMB SMB_COM_TRANSACTION Max Data Count of 0 DOS Attempt";
flow:to_server,established;     content:"|00|";     depth:1;
content:"|FF|SMB%";  depth:5;  offset:4;  content:"|00  00|";
depth:2;         offset:45;         reference:bugtraq,5556;
reference:cve,2002-0724;            reference:nessus,11110;
reference:url,www.corest.com/common/showdoc.php?idx=262;
reference:url,www.microsoft.com/technet/security/bulletin/MS
02-045.mspx; classtype:denial-of-service; sid:2102; rev:10;)
# sam.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.134 139 (msg:"NETBIOS
SMB SMB_COM_TRANSACTION Max Data Count of 0 DOS Attempt";
flow:to_server,established;     content:"|00|";     depth:1;
content:"|FF|SMB%";  depth:5;  offset:4;  content:"|00  00|";
depth:2;         offset:45;         reference:bugtraq,5556;
reference:cve,2002-0724;            reference:nessus,11110;
reference:url,www.corest.com/common/showdoc.php?idx=262;
reference:url,www.microsoft.com/technet/security/bulletin/MS
02-045.mspx; classtype:denial-of-service; sid:2102; rev:10;)
# fs.centrale.unipg.it
pass ip 141.250.0.0/16 $SHELLCODE_PORTS -> 141.250.198.140
445 (msg:"SHELLCODE x86 0x90 NOOP unicode"; content:"|90 00
90  00  90  00  90  00  90  00  90  00  90  00  90  00|";
classtype:shellcode-detect; sid:2314; rev:2;)
pass ip 141.250.198.140 $SHELLCODE_PORTS -> 141.250.199.15
445 (msg:"SHELLCODE x86 0x90 NOOP unicode"; content:"|90 00
90  00  90  00  90  00  90  00  90  00  90  00  90  00|";
classtype:shellcode-detect; sid:2314; rev:2;)
```


ISSN: 2279-8773

```
pass tcp 141.250.0.0/16 any -> 141.250.198.140 139
(msg:"NETBIOS SMB SMB_COM_TRANSACTION Max Data Count of 0
DOS Attempt"; flow:to_server,established; content:"|00|";
depth:1; content:"|FF|SMB%"; depth:5; offset:4; content:"|00
00|"; depth:2; offset:45; reference:bugtraq,5556;
reference:cve,2002-0724; reference:nessus,11110;
reference:url,www.corest.com/common/showdoc.php?idx=262;
reference:url,www.microsoft.com/technet/security/bulletin/MS
02-045.mspx; classtype:denial-of-service; sid:2102; rev:10;)

# netbios.rules
# arwen.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.131 445 (msg:"NETBIOS
SMB-DS IPC$ unicode share access";
flow:established,to_server; content:"|00|"; offset:0;
depth:1; content:"|FF|SMBu"; distance:3; within:5;
byte_test:1,&,128,6,relative; pcre:"/^.{27}/R";
byte_jump:2,7,little,relative; content:"I|00|P|00|C|00 24 00
00 00|"; nocase; distance:2;
flowbits:set,smb.tree.connect.ipc; classtype:protocol-
command-decode; sid:2466; rev:6;)
pass tcp $HOME_NET any -> 141.250.198.131 445 (msg:"NETBIOS
SMB-DS Session Setup AndX request unicode username overflow
attempt"; flow:to_server,established; content:"|00 00|";
distance:0; content:"|00 00|"; distance:0; content:"|00|";
depth:1; byte_test:2,>,322,2; content:"|FF|SMBs"; depth:5;
offset:4; nocase; byte_test:1,&,128,6,relative;
byte_test:2,>,255,54,relative,little; content:"|00|";
distance:56; content:"|00 00|"; distance:255; content:"|00
00|"; distance:0; reference:bugtraq,9752;
reference:url,www.eeye.com/html/Research/Advisories/AD200402
26.html; classtype:attempted-admin; sid:2404; rev:5;)
# fs.centrale.unipg.it
pass tcp 141.250.0.0/16 any -> 141.250.198.140 445
(msg:"NETBIOS SMB-DS Session Setup AndX request unicode
username overflow attempt"; flow:to_server,established;
content:"|00 00|"; distance:0; content:"|00 00|";
distance:0; content:"|00|"; depth:1; byte_test:2,>,322,2;
content:"|FF|SMBs"; depth:5; offset:4; nocase;
byte_test:1,&,128,6,relative;
byte_test:2,>,255,54,relative,little; content:"|00|";
distance:56; content:"|00 00|"; distance:255; content:"|00
00|"; distance:0; reference:bugtraq,9752;
reference:url,www.eeye.com/html/Research/Advisories/AD200402
26.html; classtype:attempted-admin; sid:2404; rev:5;)
pass tcp 141.250.0.0/16 any -> 141.250.198.140 139
(msg:"NETBIOS SMB-DS Session Setup AndX request unicode
username overflow attempt"; flow:to_server,established;
content:"|00 00|"; distance:0; content:"|00 00|";
distance:0; content:"|00|"; depth:1; byte_test:2,>,322,2;
content:"|FF|SMBs"; depth:5; offset:4; nocase;
byte_test:1,&,128,6,relative;
```


ISSN: 2279-8773

```
byte_test:2,>,255,54,relative,little;        content:"|00|";
distance:56; content:"|00 00|"; distance:255; content:"|00
00|";          distance:0;          reference:bugtraq,9752;
reference:url,www.eeye.com/html/Research/Advisories/AD200402
26.html; classtype:attempted-admin; sid:2404; rev:5;)
pass   tcp   141.250.0.0/16   any   ->   141.250.198.140   any
(msg:"NETBIOS    SMB    IPC$    unicode    share    access";
flow:established,to_server;     content:"|00|";     offset:0;
depth:1;     content:"|FF|SMBu";     distance:3;     within:5;
byte_test:1,&,128,6,relative;              pcre:"/^.{27}/R";
byte_jump:2,7,little,relative; content:"I|00|P|00|C|00 24 00
00           00|";           nocase;           distance:2;
flowbits:set,smb.tree.connect.ipc;       classtype:protocol-
command-decode; sid:538; rev:14;)
pass   tcp   141.250.0.0/16   any   ->   141.250.198.140   445
(msg:"NETBIOS      SMB-DS      IPC$      share      access";
flow:established,to_server;     content:"|00|";     offset:0;
depth:1;     content:"|FF|SMBu";     distance:3;     within:5;
byte_test:1,!&,128,6,relative;             pcre:"/^.{27}/R";
byte_jump:2,7,little,relative; content:"IPC|24 00|"; nocase;
distance:2;               flowbits:set,smb.tree.connect.ipc;
classtype:protocol-command-decode; sid:2465; rev:6;)
pass   tcp   141.250.0.0/16   any   ->   141.250.198.140   445
(msg:"NETBIOS    SMB-DS    IPC$    unicode    share    access";
flow:established,to_server;     content:"|00|";     offset:0;
depth:1;     content:"|FF|SMBu";     distance:3;     within:5;
byte_test:1,&,128,6,relative;              pcre:"/^.{27}/R";
byte_jump:2,7,little,relative; content:"I|00|P|00|C|00 24 00
00           00|";           nocase;           distance:2;
flowbits:set,smb.tree.connect.ipc;       classtype:protocol-
command-decode; sid:2466; rev:6;)
pass   tcp   141.250.0.0/16   any   ->   141.250.198.140   445
(msg:"NETBIOS          SMB-DS          DCERPC          LSASS
DsRolerUpgradeDownlevelServer       exploit       attempt";
flow:to_server,established;
flowbits:isset,netbios.lsass.bind.attempt;
content:"|FF|SMB";      depth:4;      offset:4;      nocase;
content:"|05|";   distance:59;   content:"|00|";   within:1;
distance:1;   content:"|09   00|";   within:2;   distance:19;
reference:bugtraq,10108;           reference:cve,2003-0533;
reference:url,www.microsoft.com/technet/security/bulletin/MS
04-011.mspx; classtype:attempted-admin; sid:2514; rev:7;)
pass ip 141.250.0.0/16 $SHELLCODE_PORTS -> 141.250.198.140
445 (msg:"SHELLCODE x86 stealth NOOP"; content:"|EB 02 EB 02
EB   02|";   reference:arachnids,291;   classtype:shellcode-
detect; sid:651; rev:8;)
pass   tcp   141.250.198.140   any   ->   141.250.199.15   445
(msg:"NETBIOS    SMB-DS    IPC$    unicode    share    access";
flow:established,to_server;     content:"|00|";     offset:0;
depth:1;     content:"|FF|SMBu";     distance:3;     within:5;
byte_test:1,&,128,6,relative;              pcre:"/^.{27}/R";
byte_jump:2,7,little,relative; content:"I|00|P|00|C|00 24 00
```


ISSN: 2279-8773

```
00           00|";           nocase;           distance:2;
flowbits:set,smb.tree.connect.ipc;       classtype:protocol-
command-decode; sid:2466; rev:6;)
pass tcp 141.250.198.140 any -> 141.250.199.15 445
(msg:"NETBIOS SMB-DS Session Setup AndX request unicode
username overflow attempt"; flow:to_server,established;
content:"|00 00|"; distance:0; content:"|00 00|";
distance:0; content:"|00|"; depth:1; byte_test:2,>,322,2;
content:"|FF|SMBs"; depth:5; offset:4; nocase;
byte_test:1,&,128,6,relative;
byte_test:2,>,255,54,relative,little;        content:"|00|";
distance:56; content:"|00 00|"; distance:255; content:"|00
00|";          distance:0;          reference:bugtraq,9752;
reference:url,www.eeye.com/html/Research/Advisories/AD200402
26.html; classtype:attempted-admin; sid:2404; rev:5;)
pass tcp 141.250.198.140 any -> 141.250.199.15 445
(msg:"NETBIOS SMB-DS Session Setup NTMLSSP unicode asn1
overflow       attempt";       flow:established,to_server;
content:"|00|"; offset:0; depth:1; content:"|FF|SMBs";
distance:3;     within:5;     byte_test:1,&,128,6,relative;
pcre:"/^.{27}/R";
byte_test:4,&,2147483648,21,relative,little;
content:!"NTLMSSP";          distance:27;          within:7;
asn1:double_overflow, bitstring_overflow, relative_offset
27,    oversize_length    2048;    reference:bugtraq,9633;
reference:bugtraq,9635;             reference:cve,2003-0818;
reference:nessus,12052;             reference:nessus,12065;
classtype:protocol-command-decode; sid:3003; rev:2;)
# frodo.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.133 any (msg:"NETBIOS
SMB IPC$ unicode share access"; flow:established,to_server;
content:"|00|"; offset:0; depth:1; content:"|FF|SMBu";
distance:3;     within:5;     byte_test:1,&,128,6,relative;
pcre:"/^.{27}/R";             byte_jump:2,7,little,relative;
content:"I|00|P|00|C|00 24 00 00 00|"; nocase; distance:2;
flowbits:set,smb.tree.connect.ipc;       classtype:protocol-
command-decode; sid:538; rev:14;)
pass tcp $HOME_NET any -> 141.250.199.133 139 (msg:"NETBIOS
SMB Session Setup NTMLSSP unicode asn1 overflow attempt";
flow:established,to_server;    content:"|00|";    offset:0;
depth:1;    content:"|FF|SMBs";    distance:3;    within:5;
byte_test:1,&,128,6,relative;              pcre:"/^.{27}/R";
byte_test:4,&,2147483648,21,relative,little;
content:!"NTLMSSP";          distance:27;          within:7;
asn1:double_overflow, bitstring_overflow, relative_offset
27,    oversize_length    2048;    reference:bugtraq,9633;
reference:bugtraq,9635;             reference:cve,2003-0818;
reference:nessus,12052;             reference:nessus,12065;
classtype:protocol-command-decode; sid:3000; rev:2;)
# sam.centrale.unipg.it
pass tcp $HOME_NET any -> 141.250.198.134 any (msg:"NETBIOS
SMB IPC$ unicode share access"; flow:established,to_server;
```


ISSN: 2279-8773

```
content:"|00|";   offset:0;   depth:1;   content:"|FF|SMBu";
distance:3;      within:5;      byte_test:1,&,128,6,relative;
pcre:"/^.{27}/R";              byte_jump:2,7,little,relative;
content:"I|00|P|00|C|00 24 00 00 00|"; nocase; distance:2;
flowbits:set,smb.tree.connect.ipc;       classtype:protocol-
command-decode; sid:538; rev:14;)
pass tcp $HOME_NET any -> 141.250.199.134 139 (msg:"NETBIOS
SMB Session Setup NTMLSSP unicode asn1 overflow attempt";
flow:established,to_server;     content:"|00|";     offset:0;
depth:1;     content:"|FF|SMBs";     distance:3;     within:5;
byte_test:1,&,128,6,relative;               pcre:"/^.{27}/R";
byte_test:4,&,2147483648,21,relative,little;
content:!"NTLMSSP";          distance:27;          within:7;
asn1:double_overflow,   bitstring_overflow,   relative_offset
27,     oversize_length     2048;     reference:bugtraq,9633;
reference:bugtraq,9635;             reference:cve,2003-0818;
reference:nessus,12052;              reference:nessus,12065;
classtype:protocol-command-decode; sid:3000; rev:2;)

# web-cgi.rules
# elearning.centrale.unipg.it
pass  tcp  $HOME_NET  any  ->  141.250.198.142  $HTTP_PORTS
(msg:"WEB-CGI calendar access"; flow:to_server,established;
uricontent:"/calendar";  nocase;  classtype:attempted-recon;
sid:882; rev:5;)

# attack-responses.rules
# tsm.centrale.unipg.it
pass  tcp  141.250.0.0/16  any  ->  141.250.198.136  1500
(msg:"ATTACK-RESPONSES directory listing"; flow:established;
content:"Volume   Serial   Number";   classtype:bad-unknown;
sid:1292; rev:9;)

# Other Rules
# frodo.centrale.unipg.it, sam.centrale.unipg.it
pass tcp 141.250.198.133 any -> 141.250.199.15 445
pass tcp 141.250.198.134 any -> 141.250.199.15 445
pass tcp 141.250.199.16 any -> 141.250.198.133 3389
pass tcp 141.250.199.16 any -> 141.250.198.134 3389
# fs.centrale.unipg.it
pass tcp 141.250.198.140 any -> 141.250.199.15 139
# gandalf.centrale.unipg.it
pass tcp 141.250.83.135 any -> 141.250.198.128 80
# X11: 'X11 outbound client connection detected'
pass tcp 141.250.199.10 6000 -> 141.250.198.4 any
pass tcp 141.250.199.10 6000 -> 141.250.198.129 any
pass tcp 141.250.199.18 6000 -> 141.250.198.4 any
pass tcp 141.250.199.18 6000 -> 141.250.198.129 any
pass tcp 141.250.199.20 6000 -> 141.250.198.4 any
pass tcp 141.250.199.20 6000 -> 141.250.198.129 any
# X11: 'X11 xopen'
pass tcp 141.250.198.4 any -> 141.250.199.18 6000
```


ISSN: 2279-8773

Appendice E – Altre script prodotte come contributi e note

[root@linux src]# **cat /usr/local/bin/chk_raid.sh**

```
#!/bin/bash
# Script che controlla l'eventuale malfunzionamento di un
array RAID-1.
# In caso affermativo invia una e-mail la quale contiene il
contenuto del file /proc/mdstat.
MAIL=/bin/mail
MESSAGE_MD=/tmp/mess_raid.tmp
RCPT_MAIL="carlo.manuali@unipg.it"
let counter=`cat /proc/mdstat |grep UU |wc -l`-`cat
/proc/mdstat |grep md |wc -l`
if [ $counter != 0 ]; then
      cat /proc/mdstat > $MESSAGE_MD && date >> $MESSAGE_MD
      $MAIL -s "RAID-1 Problem on `hostname`"
      $RCPT_MAIL<$MESSAGE_MD
fi
rm -f $MESSAGE_MD
```

[root@linux src]# **cat /usr/local/bin/chk_filesystem.sh**

```
#!/bin/bash
# Script che controlla la presenza di filesystem il cui
spazio occupato e' superiore al 95%.
# In caso affermativo invia una e-mail la quale contiene la
lista dei filesystem interessati.
DF=/bin/df
AWK=/bin/awk
MAIL=/bin/mail
MESSAGE_FS=/tmp/mess_filesystem.tmp
RCPT_MAIL="carlo.manuali@unipg.it"

$DF -Pk | grep -v Filesystem | $AWK '{if
(strtonum(substr($5,1,length($5)-1))>95) print($0)}' >
$MESSAGE_FS
```

ISSN: 2279-8773

```
if [ "`cat $MESSAGE_FS`" != "" ]; then
      date >> $MESSAGE_FS
      $MAIL   -s   "Check   `hostname`   for   out-of-space
      filesystems" $RCPT_MAIL<$MESSAGE_FS
fi
rm -f $MESSAGE_FS
```

```
[root@linux src]# cat Oinkmaster_NOTE


# tar -zxvf oinkmaster-1.2.tar.gz
# cp oinkmaster-1.2/oinkmaster.conf /etc
# cp oinkmaster-1.2/oinkmaster.pl /usr/local/bin
# cp oinkmaster-1.2/oinkmaster.1 /usr/local/man/man1

[modifying /etc/oinkmaster.conf]
url = http://www.snort.org/pub-bin/oinkmaster.cgi \
/<oinkcode.txt>/snortrules-snapshot-CURRENT.tar.gz
tmpdir = /tmp/oinkmaster
# skipfile snort.conf
skipfile unipg.rules
skipfile pass.rules
```


ISSN: 2279-8773