DRAFT

Watcher

A ‘superordinate’ firewall management solution and realtime intrusion detection system

for Linux server systems

to keep network bandits away …

Modules Manual

Revision 1.2

Contents

1 Preface

Watcher modules are the real ‘work horses’ in the Watcher system.

- They provide **real-time intrusion detection**
- They provide **firewall DROP measures** upon detection **in real-time**
- They directly track the system logger stream for a ‘facility’ and measure in **databases** (instead of slow linear file searches and writes) **for maximum performance**.

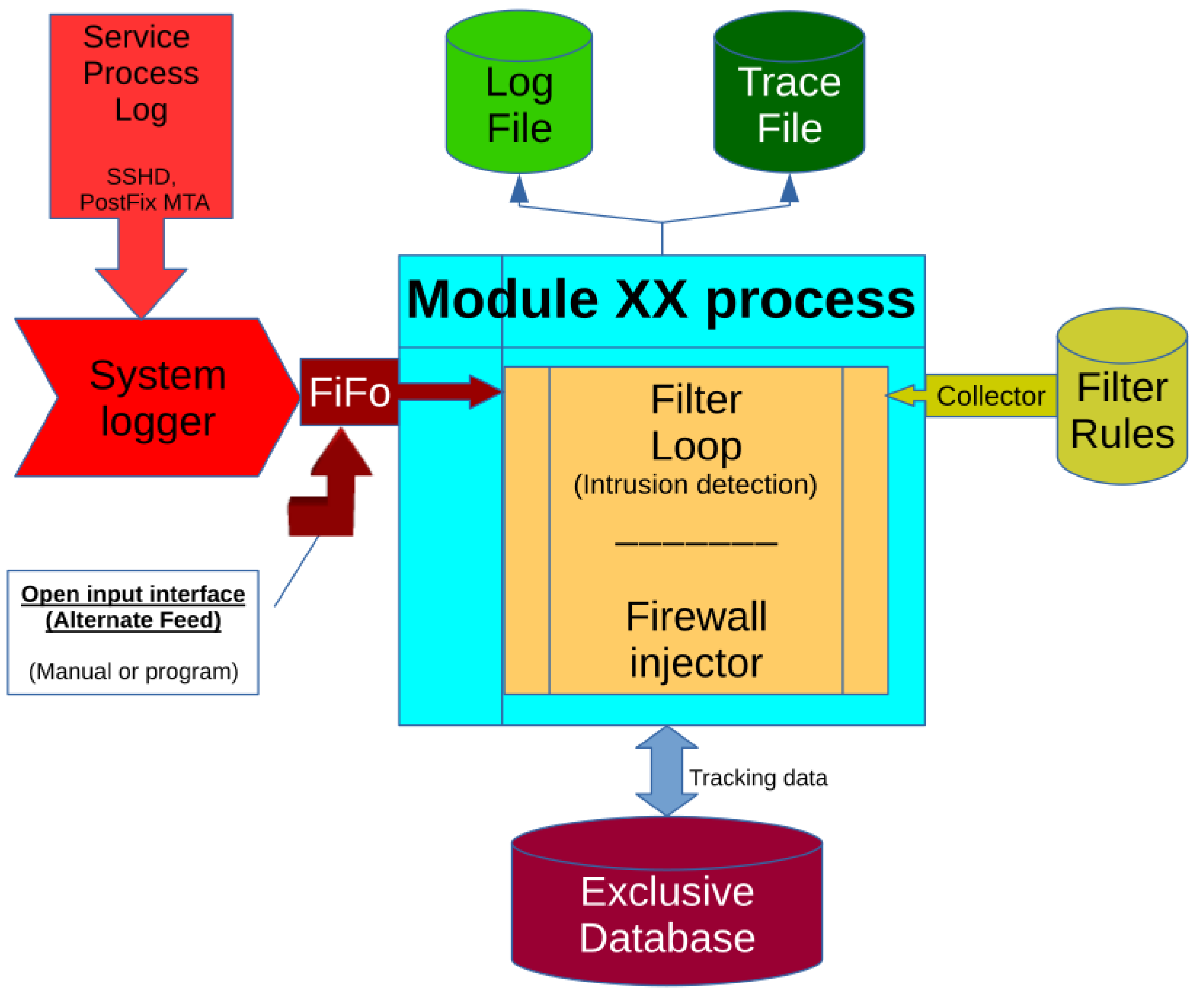
Watcher modules **run autonomously** if once started by the Watcher master service.

Furthermore each module has its individual:
- **Database expiration** tool (*ExpireXX*) for **database housekeeping** to keep the database compact for best performance
- **Statistics function** (*StatXX*) for measuring efficiency of all measure you have been taking.

Watcher modules are tracking for several services (login, mail transport, mailbox access) the **events of real attacks** for the **distinct system** instead of flooding the firewall with hypothetical lists taken from the internet.

IP addresses will be classified by the way they resolve ‘forward’ and ‘reverse’.
An IP address that does not resolve to an FQDN by any DNS is reported as ‘NXDOMAIN’ (Non-eXistent-DOMAIN) by a DNS; i.e. not registered anywhere.

1.1 Modules - Common architecture



1.2 Watcher modules

The watcher modules WatchLG & WatchMX (with its companion WatchMB with a somewhat simplified architecture) track the input to several log files on the system. Infact they don‘t read the actual log file but are fed directly by the system logger (rsyslog, syslog-ng,…) though FIFOs (‚named pipes‘) .

<u>This way the system logger can feed the Watcher modules in real-time</u>

So there are no troubles with ‚tail reading‘ of system specific files. But the system logger must be configured to write the FIFOs (‚named pipes‘) that the modules need in order to read from them. Which log files are affected on the particular system is entirely transparent to the Watcher module. The association in the system logger configuration is directly made between the system-logger's 'facility' and the module FIFO of the specific module. This makes the Watcher module system-independent and no care must be taken, whether the system is *DEBIAN*-, *SUSE*- or *RedHat*-style.

rsyslog … (/etc/rsyslog.conf)

```
...
# The authpriv file has restricted access.
authpriv.*                                  /var/log/secure
                                            |/tmp/WatchLG

# Log all the mail messages in one place.
mail.*                                            /var/log/maillog
                                            |/tmp/WatchMX
...
```

syslog-ng … (/etc/syslog-ng/syslog-ng.conf)

```
...
destination d_auth        {       file("/var/log/secure");
                                  pipe("/tmp/WatchLG");
                          };
destination d_mail        {       file("/var/log/maillog");
                                  pipe("/tmp/WatchMX");
                          };
...
```

(Restart the system logger after these changes)

1.2.1 Login watcher (WatchLG)

The 'login watcher" (WatchLG) is the simplest of all watcher modules.

It tracks the **failed login attempts** with a counter in the database for each IP address. If a number of maximum failed attempts (in the Watcher nomenclature called '*affairs*') it records 'DROP' in the database and immediately fires a DROP into the firewall to stop the aggressor instantly. The standard value for MAX_AFFAIRS is 5 – but can be changed in the WatchXX.conf file to a value of your choice.

1.2.2 Mail transport watcher (WatchMX)

The mail transport watcher tracks the input to its **exclusive FIFO in /*tmp/WatchMX***.

It is a lot more complex than the LG watcher module that only has to take care of the system's login process. WatchMX gets everything that is logged by mail related system services (PostFix, Dbmail, …) to the 'mail.*' facility of the system-logger that usually goes to the */var/log/maillog log file*.

Different mailing system (MTAs) with very different messaging to the system logger brings up the need for very individual filter rules for the specific MTA that is used on the system: PostFix, QMAIL, Exim, … etc.

In the module's delivery package the rules are configured for ***PostFix*** [MTA] and ***DBmail*** [mailbox service; POP, IMAP, ...]

1.2.3 Mailbox and SASL access watcher (WatchMB)

WatchMB is the companion of WatchMX.

WatchMB gets maillog messages **passed from WatchMX** that have to do with authentication to mailbox access via POP & IMAP (similar to 'login' tracking) or transport authentication via SASL to the MTA.

If WatchMX sees by a filter rule that this is an issue of break-in attempt the rule forwards the maillog line to the 'named pipe' that WatchMB reads for further processing in WatchMB.

Wrong passwords in mailbox access or wrong certificates in case of SSL/TLS transport requests are good examples for this.

(Rule file of WatchMX in order to forward to WatchMB)

```
RULE="Mailbox-Breaker"
      Pattern='Error\:\[pop3\]'
#--------
      result=`echo "$REPLY" | grep "$Pattern"`
      if [ ! -z "$result" ]
      then  echo  "$REPLY" >> /tmp/WatchMB
            return 2   # Flag forward action with ret-code 2
      fi
#--------

RULE=SMTPS
      Pattern='SSL_accept error from unknown\['
#--------
      result=`echo "$REPLY" | grep "$Pattern"`
      if [ ! -z "$result" ]
      then  echo  "$REPLY" >> /tmp/WatchMB
            return 2   # Flag forward action with ret-code 2
      fi
#--------
```

Note: WatchMX and the companion process WatchMB share the same database but have separate FIFOs in */tmp/...*

1.3 Modules - Common utilities

1.3.1 Database Expiration

With time the database get filled more and more with attacker’s IP addresses and DROP information that in turn will fill the firewall more and more.

Experience shows that a lot of break-in attempts are coming from a NXDOMAIN (Non-eXistend-Domain); i.e. an IP addresss that is nowhere registered by a legal Domain-Name-Service (DNS).

Keeping these addresses in the database forever is not such a good idea.

So each watcher module has an ‘ExpireXX’ program that can be started via crontab on a regular basis to cleanup the database. (where XX is the module token ‘LG’ or ‘MX’)

(crontab entries for a weekly cleanup at 00:00 [midnight] on sundays)

0 0 * * 0	<installdir>/modules/<modulename>/ExpireXX
… or ...	
0 0 * * 0	<installdir>/modules/<modulename>/ExpireXX <days>

ExpireLG has a standard value of EXPIRATIONDAYS=30 configured in the WatchLG.conf file.

To override this standard value ExpireLG can take a commandline parameter to run it with a lower value to cut the level in the database down.

To set a higher value it is recommended the increase the value in the configuration file.

The ExpireXX for the particular module dynamically removes the DROP entries in the firewall. So there is no need to restart the watcher service after an ExpireXX program ran.

1.3.2 Statistics output

Each module has a utility program to ouput ‘StatXX’.

The StatXX program generates a *.csv file from its module database that can be mailed to a configured REPORTMAIL email address. See section “Dealing with statistics files” for details.

To configure delivery of statistics data you need to configure a CRONTAB entry in the super-users’s crontab, that conducts the delivery.

(root’s crontab ...)

```
#--- Statistics   : Once a week
40 02 * * 0 cd /root/bin/Watcher/modules/WatchMX && ./StatMX >/dev/null 2>&1
50 02 * * 0 cd /root/bin/Watcher/modules/WatchLG && ./StatLG >/dev/null 2>&1
```

2 Installation manual

The watcher service takes some basic system resources and conditions that it can work.

If it comes to installed modules the most important component on the system is the system logger by which the ‘module readers’ are fed – instead of ‘tail reading’ system specific log files in /var/log/…

2.1 Preparation

In each module installation directory (modules/XXXX/... below the Watcher master path) you will find a program named ***Prep*** like in the installation directory of the Watcher master.

The *Prep* script for a module does not have much to do. But it will initialize the database from the Schema template.

Just go to the module installation directory and type in: **./Prep [ENTER]**

This will establish the exclusive database for the module.

2.2 System-logger

The modules don’t do any ‘tail reading’ or scanning of log files in /var/log/…

Instead the module processes are directly fed by the system-logger (rsyslog, syslog-ng, …) through FIFOs (‘named pipes’) located on the /tmp filesystem as:

- */tmp/*WatchLG
- */tmp/*WatchMX & */tmp/*WatchMB (companion process of WatchMX)

The benefit from this is, that log messages don’t get lost if any of the Watcher modules is going offline for a while: e.g. for database maintenance, update or whatever. The system logger continues to fill the FIFO with messages that it has picked up for a ‘facility’ from a particular service process; e.g. from the *PostFix* mail transport agent [MTA].

It the Watcher module comes back online and operational after it has been stopped for some maintenance action, then the FIFO is read just with some delay but no loss.

2.3 Logrotate

Watcher modules write individual log- and trace-files.

For housekeeping of the log- and trace-files *logrotate* configuration files should be set up. In particular if tracing is configured to be '*on*' for a module the trace output files can grow relatively quick to tremendous sizes as they track the processing of the **Real-time Intrusion Detection System (IDS).**

On RedHat-style systems logrotate configuration files are in */etc/logrotate.d.*

[root@vmd28527 logrotate.d]# tail WatchLG-.conf*

```
==> WatchLG-log.conf <==
/var/log/WatchLG.log {
        monthly
}
```

```
==> WatchLG-trace.conf <==
/root/bin/Watcher/modules/WatchLG/WatchLG.trace {
        weekly
        maxage 90
}
```

Setup the logrotate configuration files for the MX module accordingly by replacing 'LG' with 'MX' for the configuration files and inside the files as well relating to the examples above.

2.3.1 Log files

Log files are kept below /var/log/… as <*Module name*>.log

If a log file does not exist if the watcher module starts then the particular log file will be created automatically and set to R/W access exclusively for the super-user.

2.3.2 Trace files

Trace files are kept in the **module directory(!)** as <*Module name*>*.trace*

If the trace file does not exist when the watcher module starts then the particular trace file is established and set to R/W access exclusively for the super-user.

2.3.2.1 The UNTREATED rule

The UNTREATED rule is an internally hard-coded rule inside of each module code.

If none of the configured custom rules matched in the ‘filter’ function then the UNTREATED rule acts as a ‘catch all’ and outputs the log line into the module’s trace file:

Examples from an excerpt of the *WatchLG.trace file:*

1	20201115T08:02:23 WatchLG[13117]: [UNTREATED] 'Nov 15 08:02:23 vmd28527 sshd[20561]: Unable to negotiate with 139.162.247.102 port 50038: no matching host key type found. Their offer: ssh-dss [preauth]'
2	20201116T10:51:51 WatchLG[1654]: [UNTREATED] 'Nov 16 09:51:33 vmd28527 polkitd[535]: Loading rules from directory /etc/polkit-1/rules.d' 20201116T10:51:51 WatchLG[1654]: [UNTREATED] 'Nov 16 09:51:33 vmd28527 polkitd[535]: Loading rules from directory /usr/share/polkit-1/rules.d' 20201116T10:51:51 WatchLG[1654]: [UNTREATED] 'Nov 16 09:51:34 vmd28527 polkitd[535]: Finished loading, compiling and executing 9 rules'

If you find “[UNTREATED]” remarks in a module’s trace file you have 2 options:

1. Write a rule, that handles the event in the ‘filter’ function
   Note: This makes only sense if there is an IP address in the log line.
2. Drop a line into the ‘superflous_map’ file that resides in the ‘rules’ directory along with the rules.

3 Operation Manual

3.1 Writing rules

Prior to revision 1.2 the filter rules were hard-coded in each particular Watcher module and are configured for the log output of ‘Postfix’ and ‘DBmail’.

With Watcher revision 1.2 the **dynamic rule system** was introduced. This means, that rules now reside in external files and will be assembled dynamically, if the module starts.

The rule sets are still pre-configured for ‘Postfix’ & ‘DBmail’ and can now be adapted to a specific MTA and/or POP/IMAP mailbox service of choice; e.g. if the also popular mailbox service ‘*DoveCot*’ is used on a particular system.

3.1.1 Rule file format

Rules can be ‘order dependent’; i.e. a **more specific** rule has to **precede a more common** rule. Therefore rules are stored in files with a 3-digit number prefix in the range of 000 to 999:

```
[root@vmd28527 rules]# ls -1 [0-9]*
        090-Break_in.rule
        100-root-login.rule
        150-NonPriv-invalid.rule
        160-NonPriv-failed-existing.rule
```

It is possible to store several rules in a single rule file as long as the order of rules follows the ‘*more specific rule before more common* rule’ requirement is taken into account.

```
[root@vmd28527 rules]# cat 100-root-login.rule
RULE=root-login
        Pattern=": Failed password for root"
#---------------
        result=`echo "$REPLY" | grep -E "($Pattern)"`
        if [ ! -z "$result" ]
        then    : echo "--------- Matched rule $RULE ------------"
                inject
                return $?
        fi
```

3.1.2 Testing new or changed rules

The ‘rules’ directory contains two small scripts ‘*check-rule*’ and ‘*check-all-rules*’, that you can run on the ‘rules’ directory in order to check, that a rule is syntactically clean or all your rules are syntactically clean. It is strongly advised that you use this syntax-check as the rules are assembled into the ‘filter’ function by the ./mkfilter script and then the generated ‘filter’ function is sourced by the module. With syntactically wrong rules the module might fail starting or can behave erratically.

3.2 Dealing with statistics files

Each module provides a statistics script ‘StatXX’.

The module creates an output from your module’s database contents as a relation of ‘introduced’ (detected) and ‘dropped’ IP addresses in the time range that the expiration process of the module has left.

The statistics script creates a *.csv file that is sent to the configured report mail address covered by the $REPORTMAIL variable. This variable is set to a global REPORTMAIL variable in the watcher master configuration ‘watcher.conf’ and can be overwritten in the module’s WatchXX.conf file located in the module path. For instance, if you (as the systems administrator) not interested in the statistics of attacks on the mail system but the mail system administrator wants to see efficiency of the firewall related with the mail system attacks.

To get statistics mailed setup a CRONTAB entry in the super-user’s crontab:

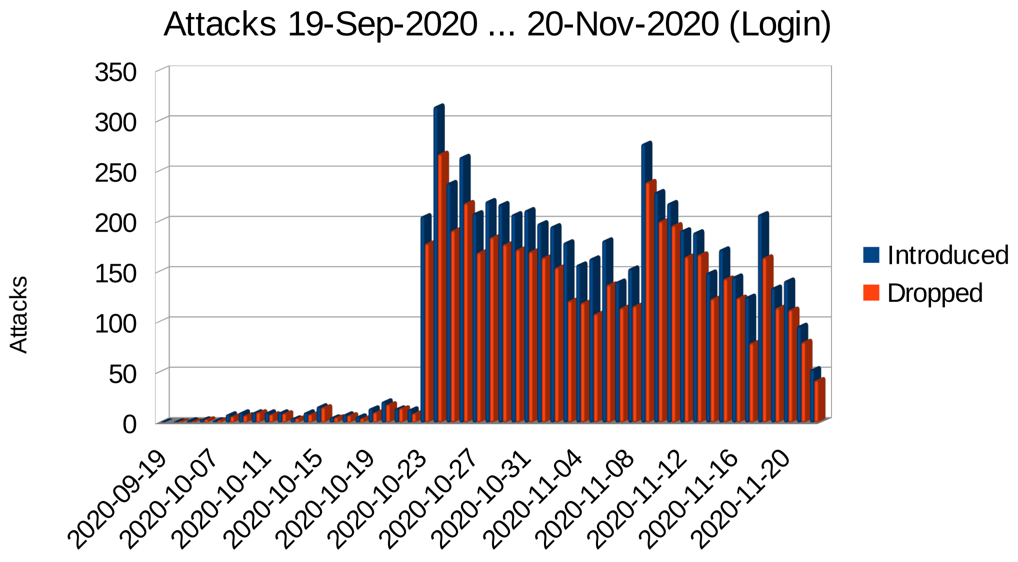
```
#============ Watcher ========================================
...
#--- Statistics  : Once a week
40 02 * * 0     cd /root/bin/Watcher/modules/WatchMX     && ./StatMX >/dev/null 2>&1
50 02 * * 0     cd /root/bin/Watcher/modules/WatchLG     && ./StatLG  >/dev/null 2>&1
```

The configured recipient will then regularly find a Statistics file in *.csv format in his mailbox with the subject: “Statistics-XX <a timestamp>”

By clicking on the attachment your favorite spread-sheet program should open and offer the CSV file to be read into a new spread-sheet.

After the statistics data was read-in you may then select from the ‘diagram functions’ the creation of a diagram of your choice.

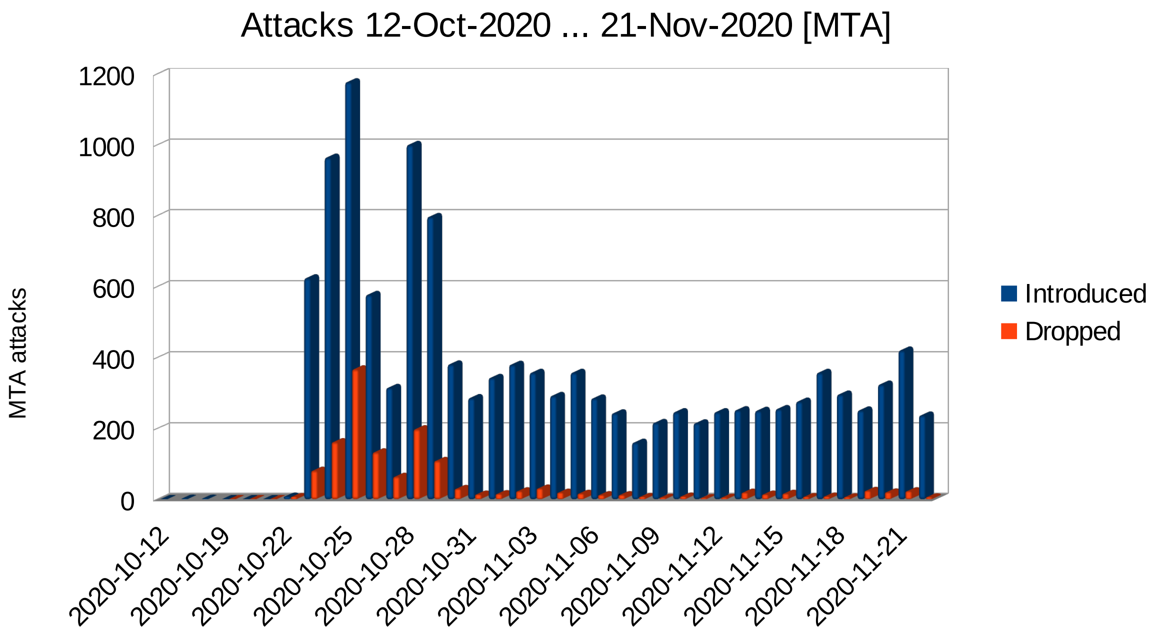
You may then annotate the diagram to your taste, include trendlines, calculated average and so on. Finally you may save and/or print the document and store it to review the efficiency of your measures.



Sample statistics diagram (Login, LG module data)

3.2.1 Interpreting statistics diagrams

Below is a statistics diagram with data from the MX (mail) module's database.



(Sample mail attacks diagram)

The diagram looks a bit odd at first view, since it shows at tremendous number of 'detects' (*introduced*) with a relatively little number of firewall DROPs – but this is fairly normal for mail attacks. SPAMers tend to use the scheme '*fire-and-forget*'; i.e. they fire their attempt once against a mail server and never come back as legal MTAs usually do to retry the mail transfer. Although NXDOMAINs (and FAKEHOSTs as well) get a preset of *MAXAFFAIRS-1* that results in a firewall DROP on second attempt there is no second attempt that would cause the DROP in the firewall for the incoming IP address. This is why the ‘introduce’ values are so high.