

User Authentication at the Firewall level

Éric Leblond, Vincent Deffontaines and Sebastien Tricaud

INL
101/103 Bvd MacDonald
75019 Paris, France
eric@inl.fr / gryzor@inl.fr / toady@inl.fr

Abstract. This paper focuses on how firewalls can work at the TCP/IP network layer and handle a user authentication where the IP address is not considered at all. We will first explain the common weaknesses of existing identity-based filtering systems, detail what exists in Netfilter internals to respond to it, and propose a user friendly implementation through the NuFW [13] project. We will conclude with some usage example of latest Netfilter features.

Key words: Netfilter, NuFW, SIEM, Firewall, Authentication

Introduction

Motivation of identity-based filtering

When defining security policy, the security officer differentiates users at the access level. Each user has a specified role in the company. And each role had its own access, whatever the person behind. The need for this differentiation is crucial because the security policy is mainly about constraints on members behavior in the organization.

However, when it comes to the technical implementation of those security policies, the administrator is limited by the tools provided with Operating Systems. If our payroll server is on the IP 192.168.1.42, port 80 and the payroll officer has the IP 192.168.66.5, the following iptables command must be run:

```
# iptables -A FORWARD -s 192.168.66.5 -p tcp -d 192.168.1.42 --dport 80 -j ACCEPT
```

And the same kind of line must be written for each user.

Since the source IP is and will never be the reliable way to make sure we are dealing with the right person, the logical answer is to perform authentication at the user level.

Existing solutions

The existing and widely-used solutions to resolve these needs are:

- Static IP/user binding
- MAC/IP/User static binding
- Dynamic IP/User binding
- 802.1x

Static IP/user binding One may find convenient for an administrator to differentiate users by giving them dedicated IP addresses and setting firewall rules based on this permanent association.

This is a totally insecure method as any system can choose its IP address. One often argues that it is not possible to steal the IP of a connected computer. This belief usually comes from how Microsoft Windows operating systems test if an address is already used before accepting to use an IP address.

But this is not the case of all operating systems, and definitely not a standard.

Furthermore, any script kiddie is able to steal the IP of a connected computer by using a dedicated program. One can use for example arpspoof provided by dsniff [11] to take over the IP of a connected computer:

```
# arpspoof -t target host
```

The method used by this software is to continuously send ARP request to pretend our computer has the host address. By doing this, ARP announcement from the host system will always be older than our own announcement and this will force all computers in the network to suppress the host computer from the ARP table in favour of ours.

MAC/IP/User static binding This solution uses the same method as the previous one, but try to enforce the security by adding the MAC address to the association.

The problem of this method is that it gives some people a security feeling. They often sadly ignore that a MAC address can also be changed without serious difficulty. For example, ifconfig or macchanger [10] are able to change mac address:

```
# macchanger --mac=01:23:45:67:89:AB eth1
```

Dynamic IP/User binding This is the most dangerous method because it is provided by many manufacturers and thus widely used:

- authpf
- Netscreen
- Kerio WinRoute firewall

All systems use a different means to dynamically find or be noticed that a given user is on a given computer. Once the association is made, it is valid for a certain amount of time or has to be kept alive by some session mechanism.

The word “dangerous” has been carefully selected at the start of this section. A post (see [9]) on bugtraq mailing-list has shown that Netscreen authentication is truly based on IP. This implies that all users behind a NAT gateway are seen as one user. The main danger comes here from the lack of documentation in Netscreen product concerning this issue. But, it could also be fast and easy for an attacker to set-up a NAT gateway and to put a innocent user behind it to steal his identity.

Even more dangerous is Kerio authentication (see [6]) which implements an almost persistent user/IP association. If a user logs onto a box, his rights are associated to all packets coming from the IP as long as there has been a packet coming from this IP in the last 2 hours. In production systems, this really looks like eternity.

This is even worse when looking into details: It is enough to use the authentication on the HTTP proxy to initiate the User/IP association. Thus, if an administrator uses the proxy on a Terminal Server, all the users are granted administrator permissions by the firewall for an almost endless time when they emit packets to the network.

802.1x 802.1x lays on a cooperation with switch equipments to authenticate a computer before forwarding any packet trough the port. Its main goal is to prevent unwanted computers to access the network. It is based on an authentication which occurs as soon as the computer tries to connect. A frequent usage is to select the destination VLAN of the computer after it has successfully authenticated. This provides a basic access control by managing afterward inter-VLAN security. But it is far from

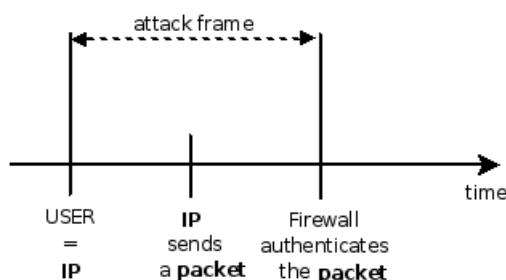
finer grained filtering rules. Furthermore, it is a mono-user solution and does not provide any means of differentiating simultaneously connected users.

One of the most easy attack against 802.1x is to put a hub between a “victim” computer and the switch. After authentication of the victim, it is easy for the attacker to set-up his system (same mac address, different IP) to access the network.

Time limit attack

With most time limit capability firewalls, the connection created by a user is never cut. Only new connection cannot start. If the firewall disallows connections from Friday 6pm to Monday 8am, and the user do not log off, the connection will remain and the firewall will still allow the connected users.

Timeout attack



Almost all described systems suffer from timeout attack. Sessions have to be kept alive to maintain the User/IP association. Since the session must exist before a packet can be assigned to a user, the user authentication is done when the session starts. Packets are authenticated “A priori”, because no further action is required from the emitter to authenticate datagrams after it is emitted.

Thus, there is a timeout system which can in most cases (if not all) be used to fake the system.

Detecting Security policy abuse

The user authentication at firewall level can also be used to detect security policy abuses. To understand more easily, we will take a fictional example:

Let suppose we are the security administrators of a bank (RF). One of the trader (IJ) is using the application login of colleagues to do operations to hide operations to the internal control system of the bank. He connects to the application from a Citrix server.

Now the question is: how can we detect this abuse ?

From the application point of view, the operations are clean: the authentication has been checked and there is no problem with the operations by itself. We can't find anything weird if we only analyze the application logs by themselves. If we try to do a analysis of application logs and Citrix session logs, we will only be able to compare the user given to the application with the list of users connected to the Citrix at the moment of the connection. If IJ is smart enough (or lucky enough), he will only steal identity when the user is connected. We will then have a match between the application user and one of the users connected to the Citrix server. Thus, we don't have a good detection system.

But if RF is using a identity based filtering policy, things are simpler. The firewall logs the IP connection and the application receives this connection. It is then easily possible to see that the provided users don't match. Once the attach is detected, we can stop the trader before its too late and avoid that RF loose 5 billions euros.

Summary

Because all existing solutions are either subject to severe security problems or not applicable on multiuser environments.

A strict control is wanted: the 2007 CSI survey [2] shows insider attack or misuse is the most frequent attack with 59% of companies reporting such an incident in the previous year.

All these “authentication” methods fail miserably, mainly because they attempt to bind users identity to objects that are not relevant to authentication. The OSI model for TCP/IP is simply not built to provide authentication in its low layers, and trying to bind a user’s identity to a MAC or IP address makes no sense in this regard.

What Netfilter provides

Introduction

Netfilter [12] is the packet filtering framework inside the Linux 2.4.x and 2.6.x kernel series. This is the successor of ipchains the Linux 2.2.X implementation of packet filtering.

The Netfilter connection tracking enables the stateful firewall feature. This feature allows firewall administrators to simply define the wanted connection and then accept any related connection. This divides the rulesets number, and with the appropriate modules, makes Netfilter able to investigate deeper in protocols to mark connections related, when this is a protocol requirement.

While it is easy to understand how Netfilter is stateful with TCP connections, Netfilter has the ability to do it for UDP. In the Linux kernel source file *linux/net/netfilter/nf_conntrack_proto_udp.c*, two variables are defined:

```
unsigned int nf_ct_udp_timeout = 30*HZ;
unsigned int nf_ct_udp_timeout_stream = 180*HZ;
```

The first variable, **nf_ct_udp_timeout** defines when we decide a connection is finished if the packet got no answer (IPS_SEEN_REPLY_BIT) and the second **nf_ct_udp_timeout_stream** is the stream timeout about answer to a previously sent packet.

The unit of there variables is the number of interruption of the timer. **HZ** constant is the scheduler frequency defined in the CONFIG_HZ constant at kernel compile time. It is the number of interruption a second from the timer. In real life unit, first timeout is 30 seconds and second is 3 minutes.

In the case of FTP, the Netfilter conntrack ftp module will mark connections as related by waiting for the FTP PORT command, that will establish the data tunnel. This may be active or passive, according to the FTP description. Helper modules are available for the mainly used protocols and among them H323, SIP, IRC, TFTP, Netbios Name service...

In the 2.6.14 Linux kernel, Netfilter developers have introduced a new system called Nfnetlink. The main goal was to improve interaction with userspace. This system has brought three components:

- **NFQUEUE**: Netfilter can forward the verdict decision to any user application connected to the NFQUEUE socket.
- **NFNETLINK_CONNTRACK**: Userspace can query or modify connection tracking entries.
- **NFLOG**: Improved logging subsystem.

The three components are subsystems of a single netlink interface. The architecture is the following:

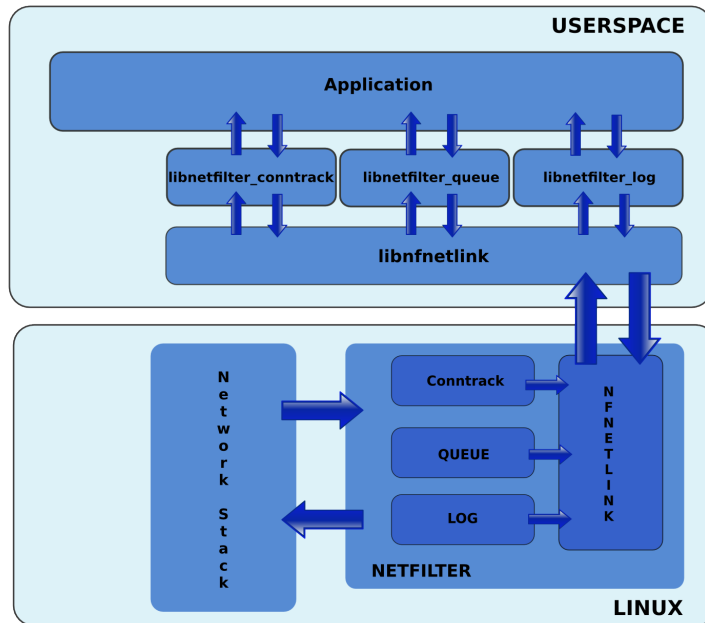


Fig. 1. Nfnetlink architecture

User space decision (NFQUEUE)

NFQUEUE provides to Netfilter the ability to forward its verdict decision to any user application connected to an NFQUEUE socket.

An example of this NFQUEUE usage is the well-known Snort program, which can be compiled as inline mode and accept or drop packets according to a match from its rules. Making Snort an Intrusion Prevention System (IPS).

This functionality was already available in Netfilter via the ip_queue system. The rewrite has brought some interesting improvements such as:

- **Multiple queues:** it is now possible to run different application which will listen for kernel message on separate netlink socket.
- **Marking capability:** userspace can put a mark on packet. This mark can be used by other network subsystems (routing, QoS) to classify packets.

Stateful firewall (NFNETLINK.CONNTRACK)

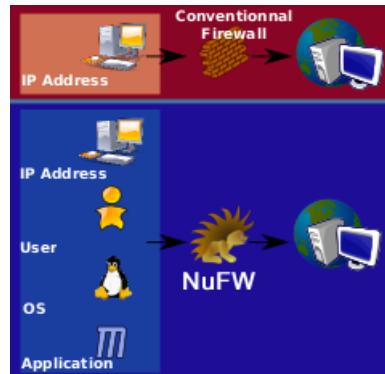
Before Nfnetlink, the only interaction with Netfilter connection tracking was a file in the /proc directory which enabled userspace applications to display the connections list. NFNETLINK.CONNTRACK introduces a complete set of actions:

- **LIST:** List connections.
- **UPDATE:** Update parameters of a given connection.
- **CREATE:** Create a connection.
- **DESTROY:** Suppress a connection from the connection tracking table.
- **LISTEN:** Listen to events such as creation or destruction of connection tracking entries.

With that system, the administrator can watch over the activity of its firewall and interact at will with Netfilter connection tracking. The main project using these features is conntrackd [1] which is a connection tracking replication daemon which enable highly available scenarios.

NuFW implementation

With NuFW, we provide a strict authentication to firewalling. This means there is no “IP==user” or “MAC==user” binding at all. NuFW works at the connection level. Every connection needs to be authenticated by their emitting user. This step is performed at the opening time for each connection.



The idea behind NuFW is very simple: adding the notion of UserID to the existing filtering criteria. This is different from other “authenticating” firewalls, such as authpf, because NuFW never makes any “IP == user” equivalence. The UserID of each connection is checked and validated strictly, so that no spoofing is possible. This means, practically, that NuFW also works with TCP/IP clients on multi-user systems: traffic from each user is recognized and signed.

NuFW requires an Agent on the client computer, in order to do the initial authentication (against the LDAP server, for example), and to authenticate connections coming from the user working on the client host.

NuFW does not modify the packets, and does not rely on information sent by the client to allow the connection : even if the client returns a faked connection list, it will have no effect on the evaluation of the filtering rule (ACL).

Starting from this point, we decided to find a strict method to authenticate user flows and we came to an “A Posteriori” authentication of each individual connections (see [8]). Afterward, these ideas have been implemented in NuFW.

How NuFW enhances the implementation of security policies

NuFW has the following functionalities and advantages:

- IP level 3 firewall log is marked per user. No need to search for “Who owned this IP address 3 months ago?”
- Filtering is more strict: NuFW implements per user IP filtering, no matter if a user shares their workstation with others.
- Server monitoring: on a UNIX webserver, root user may connect to HTTP servers on the internet to download updates (apt-get), while we want bells and whistles if some Apache process tries to open the exact same connection.
- Single Sign On.
- Apply a specific policy on the packet depending on the user: one goes directly to internet, the other is sent to a proxy.
- Direct link between user directory and filtering rules. This perfectly suits the organisation structure.

A brief chronology

NuFW has grown with the following steps:

- 2001-2004: proof of concept, no crypto on exchanges.
- 2005:
 - NuFW 1.0 - First usable, stable release.
 - NuFW wins security category of "Trophées du libre"
 - NuFW is used by a french hospital
- 2006:
 - NuFW 2.0 - Many "linting" options: daemons support reload, send ICMP datagram when rejecting a connection, Prelude IDS [3] logging support, Time-based ACLs support, user session logging.
 - NuFW is used by NATO
- 2007
 - NuFW 2.2 - IPv6 support, Support for per user routing and QoS, Protocol enhancement, Command mode for interactive administration.
 - Gendarmerie Nationale (part of military department) uses NuFW
 - NuFW wins innovation price of "Lutèces d'or"
- 2008: French education department will deploy NuFW in every French school

NuFW algorithm

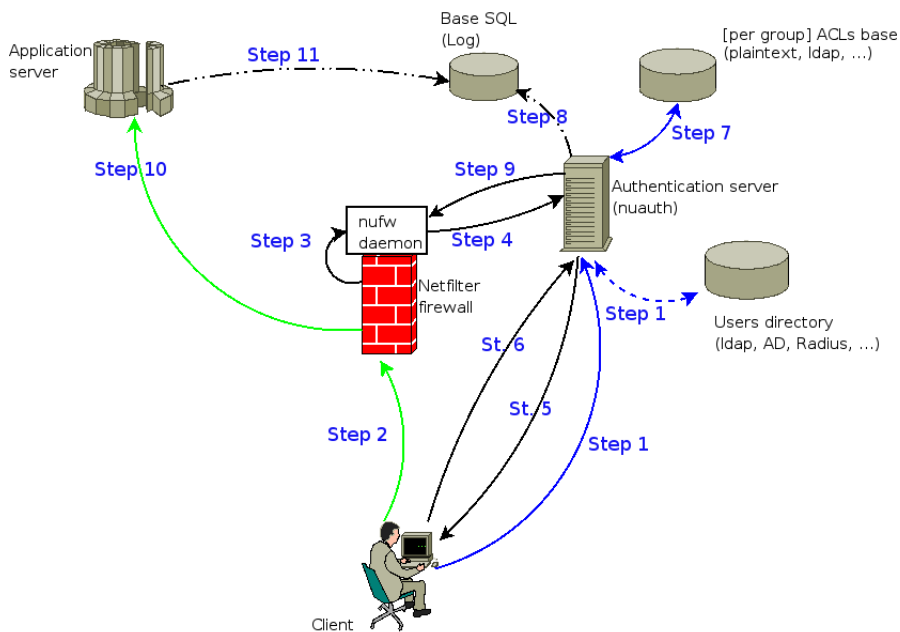


Fig. 2. NuFW algorithm

Steps with blue arrow do not need to occur on every connection. They either occur once and for all, at the start (step 1), or "once in a while" because of a cache system (step 7). Steps 8 and 11 are totally optional.

1. NuFW agent, running on client computer, opens a connection to the nuauth server. This connection lays on TLS and is only accepted by nuauth if the client provides valid credentials. This connection remains opened as long as the user/client is alive and will be used in further steps. Typically, users can only open this connection if they have a valid account in the users directory. As a posix step, nuauth fetches user's groups.
2. Client tries to connect to a resource behind the firewall. This is the datagram that initiates a connection (typically, TCP SYN).
3. The Netfilter rule matching the datagram sent at step 2 has a NFQUEUE target: decision is out of the scope of Netfilter. Netfilter will wait nufw's answer to apply it to the datagram.
4. The nufw daemon sends information about datagram sent at step 2, and requests that a decision be taken by nuauth for that datagram.
5. The nuauth daemon:
 - a. Looks at the Source IP address of the datagram sent at step 2.
 - b. Looks for all connections opened at step 1 coming from the same source IP address (there may be several if the client system hosts several processes/users with different UserID).
 - c. Sends a "Refresh request" to all connections found in b/. It uses connections opened at step 1 to send these requests.
6. Each agent that receives this refresh request looks locally (netstat API) for connections sent by the system user they run on (for TCP, connections in SYN_SENT state). Each agent sends this information to nuauth, through the connection opened at step 1. At this step, traffic is identified/authenticated: nuauth knows who sent the datagram at step 2.
7. Nuauth checks whether the user who sent the step 2 datagram is authorized to do so.
8. Optionally, nuauth logs the event to external storage (SQL, Prelude, syslog...).
9. Nuauth tells nufw what the decision for step 2 datagram is. Nufw transmits it to Netfilter that applies it.
10. If the decision is an accept, datagram sent at step 2 is transmitted to the server. It is to be noted this occurs only for the opening datagram of each connection. The flow of the connection is directly accepted by Netfilter's stateful conntrack. This, practically, means that NuFW has a proven zero impact on connections' bandwidth.
11. Optionally, if the server needs auth for this connection, it can fetch it without bothering the client from the Log database through a request such as

```
SELECT username FROM log WHERE source_ip=192.168.1.0 \
  AND source_port=2327 AND destination_ip=192.168.33.3 \
  AND destination_port=80 and protocol=6 and state=ESTABLISHED;
```

This is strict because, on TCP and UDP, one single client socket using the four parameters (source IP, source port, destination IP, destination port) can be opened at any moment. This step means "Protocol independent Single Sign On for TCP and UDP".

Performances

NuFW has an impact on the opening time of authenticated connexions. On average systems (laptop), our benches show that this impact is about 10 milliseconds per datagram with less than 2000 new connections/second. On bigger server (IBM PPC 64) we got to 4000 new connections/second. We insist that this overhead is a charge for connexion opening datagrams (for TCP, SYN packets only) ; the flow of the connexion is handled 100% normally by Netfilter's conntrack.

This is suitable for most organisations, and the NuFW team is working to raise these limits as high as possible (see roadmap).

NuFW stability

NuFW is considered stable since the release of the 1.0 branch. The current stable branch is 2.2. NuFW has been used on production installations, with several hundreds of users, since 2005.

Caveats: things to be careful about

NuFW is sensitive to NAT if it occurs between the client and the firewall. This makes NuFW relevant for authenticating connections opened by an internal client, not Internet clients. Of course, NuFW also works inside VPNs (ie, for road warriors).

UDP is complicated to authenticate (client side). For instance, on Linux, `/proc` entries do not mention what the destination of the datagram is. Authentication of UDP connections is doable, but will probably require administrator privileges on client side to keep track of UDP flows. Some protocols, like ICMP cannot be authenticated as the packets are sent by the kernel. At the time of the writing of this article, NuFW supports TCP and UDP.

Some systems use different users to perform some network tasks. Windows XP, for instance, performs DNS requests through the `svchost.exe` service, which is run by “System”, not the “real” user. Network sharing protocols, which have their connections opened by the kernel (NFS, SMB, ...) cannot be authenticated strictly, because they are opened by the client’s kernel. We actually see that such connections can often be shared by the Operating System for simultaneous users.

Using Netfilter latest extensions

Authenticated conntrack

Once upon a time, NuFW was only a packet filtering enhancement. After some time, we have implemented a strong user authentication cross-platform, stable, with log facilities, secure communication between each agent and we also improved it to do identity-based quality of service. This last one was achieved by a light kernel patch which adds to Netfilter the ability to put a mark on packet from userspace. Under Linux, the mark can be used by various subsystem to decide of the fate of a packet. NuFW is only able to put the mark on the first packet of a connection and this was not really useful for QoS. But, with the help of the `CONNMARK` [7] feature, it was easy to propagate the mark (and thus the UserID information) during all the lifetime of a connection.

Although it was not a security related issue, we thought that it could permit to retrieve some interesting usage information. But more interaction with `NFCONNTRACK` were needed to get data.

In Linux 2.6.14, Netfilter was modified to add a complete messaging system relative to `NFCONNTRACK`. This message subsystem allows userland to retrieve, insert and alter the connection tracking table. By using this new feature, it was then possible to fetch all connection tracking information and to link them with the identity information provided by NuFW. We’ve added to NuFW the ability to listen to `NFCONNTRACK` event and to send information to the logging subsystem. It was then possible to duplicate and archive connection tracking in userspace. If a connection has been authorized by NuFW, the logged entry is authenticated. This information flow is described in the following figure:

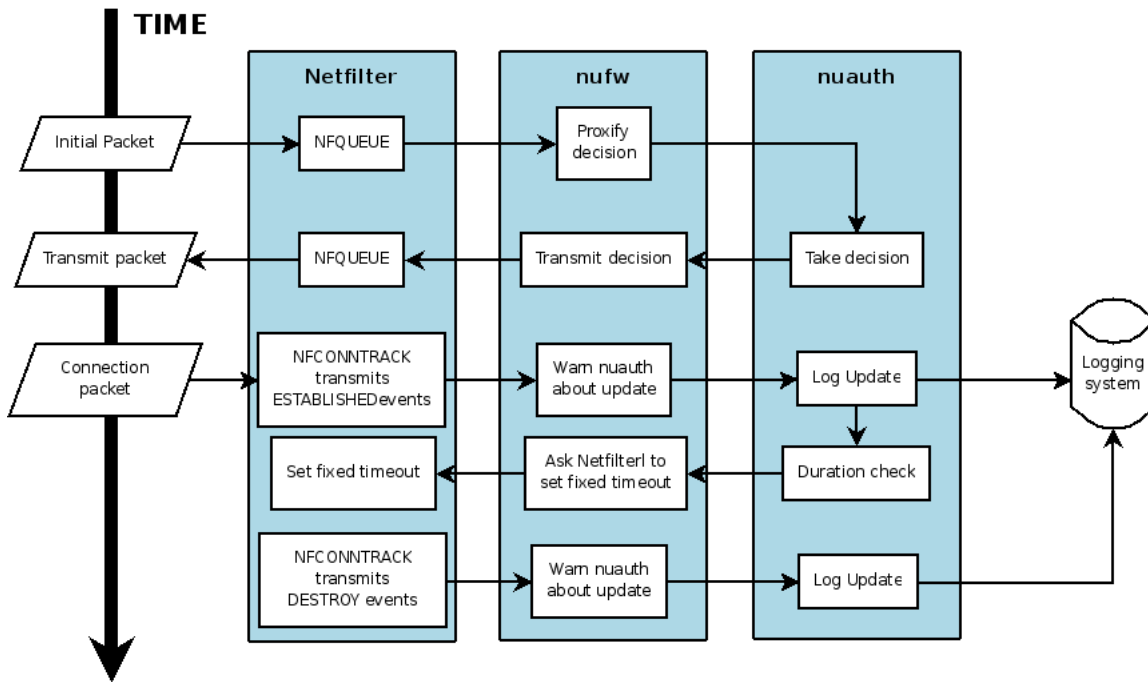


Fig. 4. NuFW scheme with time based rules

Conclusion

NuFW has succeeded in bringing the concept of users and groups into IP level 3 filtering / firewalling. It is used to strictly implement security policies, and to keep track and history of authenticated network flows attempts, be they allowed or not.

Yet, it offers a new field of applications that are to be imagined. For example, a NuFW user has thought of linking the firewall rules to the request tracker: An admin can only connect to a distant server if a ticket has been opened for it. In fact, by combining a strict authentication of packets and the power of userspace requests, NuFW can implement security policies that were previously unthinkable.

Appendix: NuFW availability

NuFW is released under the GPL license, as well as Web based interfaces we have written to ease administration (Nuface [4], Nulog [5]). NuFW agents for Free OSes are released under the GPL. NuFW agents for proprietary OSes (MS Windows) are distributed on a proprietary license.

NuFW runs on Linux firewalls (prefers 2.6, can run on 2.4), on top of Netfilter (with the QUEUE or NFQUEUE target).

NuFW authors have been contributing code to the Netfilter project ¹ for better interactions of Netfilter with userspace (conntrack modifications through libnetfilter_conntrack, fixed timeout, ...).

References

1. Pablo Neira Aruso. Conntrackd. <http://www.netfilter.org/projects/conntrack-tools/index.html>.
2. CSI. 2007 csi survey. Technical report, CSI, http://www.gocsi.com/forms/csi_survey.jhtml, 2007.
3. Prelude IDS. Prelude ids. <http://www.prelude-ids.org/>.

¹ Contributions to Netfilter: <http://software.inl.fr/trac/trac.cgi/wiki/Contribs#Netfilter>

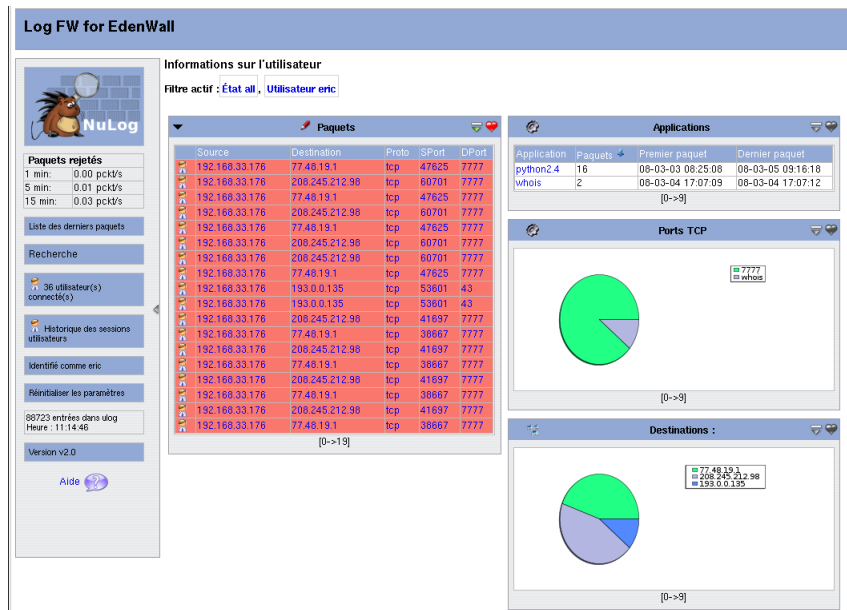


Fig. 5. Nulog, log analysis interface

4. INL. Nuface. <http://software.inl.fr/trac/trac.cgi/wiki/EdenWall/NuFace2>.
5. INL. Nulog. <http://software.inl.fr/trac/trac.cgi/wiki/EdenWall/NuLog2>.
6. Kerio. *Kerio Winroute administrator manual*. Kerio, <http://download.kerio.com/dwn/kwf6-en.pdf>, 2007. page 126-129.
7. Éric Leblond. Netfilter connmark. http://home.regit.org/?page_id=7.
8. Éric Leblond, Vincent Defontaine, and Xavier Desurmont. Efficas. <http://www.nufw.org/eficass/>, 2003.
9. Netscreen. Authentication vulnerability in netscreen screenos. <http://seclists.org/bugtraq/2003/Jun/0218.html>, June 2003.
10. Alvaro Lopez Ortega. Macchanger. <http://www.alobbs.com/macchanger>.
11. Dug Song. Dsniff. <http://www.monkey.org/~dugsong/dsniff/>.
12. Netfilter Core Team. Netfilter. <http://www.netfilter.org/>.
13. NuFW Devel Team. Nufw. <http://www.nufw.org/>.