Summary of the netfilter developer workshop 2003

Harald Welte

1. Introduction

foo

2. Topics

2.1. pkttables

pkttables is the “next generation” packet filter for linux 2.6.x and beyond.

pkttables is a whole framework for layer3 independent management of packet filtering rules. The idea is to reduce the code replication between ip_tables.c, ip6_tables.c, arp_tables.c on the kernel side, as well as the userspace counterpart (libip4/6tc, iptables.c/ip6tables.c). They all have to deal with the same set of problems: communicating packet filter rules between userspace and kernel, add/remove/replace/insert rules into existing rulesets, and matching a particular packet against a ruleset. Especially the management functions like ruleset loading from/to userspace is not dependent on the l3 protocol at all - but still we replicate this functionality in the current 2.4.x packet filter(s).

The major parts of pkttables are

1. pkt_tables - the in-kernel pkttables core
2. pkt_tables_ipv4 - the ipv4 incarnation of pkt_tables
3. pkttable_ipv4_filter - the ipv4 filter table instance
4. pkttnetlink - the in-kernel part of a netlink-based userspace interface
5. pktt_foo.c - the in-kernel part of a match extension
6. pktt_FOO.c - the in-kernel part of a target extension
7. libpkttnetlink - userspace library for low-layer communication
8. libpkttables - userspace library providing high-layer API for apps
9. pkttables - commandline program for ruleset manipulation by the admin

During the workshop, the following requirements/comments on the current pkttables design were made:

1. Use TLV’s for the data protocol used in all netlink messages. This ensures future compatibility with netlink2 (see the related IETF forces draft), and allows applications like saving/restoring iptables rules in binary

architecture-independent form. IT also introduces a clear distinction between the in-kernel representation of [matchinfo] data - and the communication structures used between kernel and userspace. As a by-product it solves any kernel64/user32 issues on SPARC64 and other similar architectures. Also, versioning can easily be implemented, new fields can easily be added, ...

2. Transactions. The goal is to make everything transaction based. However, full support for an arbitrary number of transactions is difficult to implement. As a minimum requirement: Only one open transaction per time. Once the transaction is started, all operations are performed on a copy of the ruleset. Once the transaction is committed, the rulesets are atomically replaced.
3. Versioning. Versioning is handled via sub-tlv's of the matchinfo tlv. Every plugin can define it's own sub-TLV's within the matchinfo TLV. This way, they can invent new TLV's on their own. A new version of a particular plugin would introduce a new Tag. However, both kernel and userspace still support the old tag. To find out, which tags are supported on both sides, a discovery mechanism needs to be in place.
4. libpkttables core can do all the encoding if plugin specifies the tag for a given parameter. pro: easy. con: how to register different tlvs depending on kernel version. idea: include min/max
5. matchinfo tlv tags are allocated at patch-o-matic inclusion time. This way we have a central way of assigning the numbers and assuring no overlap (NANANA). We also specify a range of experimental tags for external modules or modules that are not included in patch-o-matic yet.
6. i18n. Since libpkttables returns also the descriptions/help messages for individual plugins, support for i18n should be included in the architecture
7. manpage generation. It was noted that this could be done automatically, if all the data is hidden within libpkttables.
8. seperate target/policy. This makes it explicit on whether a target terminates or continues traversal within the chain/table. Currently we have terminating and non-terminating targets. Apart from documentation and/or experience of the user, there is no way of telling them apart. In pkttables, every target tells the core which policies (PKTT_ACCEPT, PKTT_DROP, PKTT_CONTINUE, ...) it supports. The user then has to specify a '-P ACCEPT' option on every rule in order to specify the desired behaviour.

As for kernel inclusion, it was decided that pkttables will be included as additional netfilter subsystem (in parallel to the current iptables code) during the early 2.7.x kernel series - and then backported into 2.6.current at that time.

2.2. Wallfire

Herve Eychenne gave a presentation about the wallfire project. Please refer to his slides and/or the wallfire documentation to learn more about it. This summarry will just cover the issues related to netfilter/iptables development.

- verbose error reporting (netlink messages)
- slow rule manipulation. Incremental changes to the ruleset are way too slow. This should be improved with iptables-1.3.x (including libiptc2) and the new mark_source_chains() implementation. A real 'fix' is only an incremental kernel/userspace interface, like pkttables will introduce.
- converter to convert shell script into iptables-save format (shellscript?)
- MASQUERADE: don't flush at ifdown, but at ifup (and cmp to old address). The current flush-at-ifdown policy deletes all NAT mappings, even if our ISP gives us the same IP again.

2.3. Transparent Proxying

Balazs Scheidler gave a presentation about the transparent proxying patches he has been working on. Below is a short summary of the presentation.

TPROXY has real transparency requirements. This means not only hiding the proxy to the requesting client, but also to the server. That in turn requires being able to initiate connections form a foreign source address.

The current implementation uses a separate table called 'tproxy' to perform REDIRECT-like transpaency. A target called 'TPROXY' is allso provided. The reasons are:

- NAT remains to be useful for forwarded traffic; NAT and proxy rules are cleanly seperated
- The TPROXY target marks sessions using a bit in conntrack->flags (new structure member) to make it easier to match tproxied packets in the filter table
- special handling of UDP is necessarry

The other two transparency cases are implemented using a hashtable of sockrefs. This hashtable is manipulated from userspace using setsockopt calls (IP_TPROXY_* and friends). The most important parts of sockrefs are:

- local IP:port
- foreign IP:port
- flags indicating connection mode (LISTEN or CONNECT)

When a NEW connection is processed by the TPROXY hook, it consults the sockref hash, looks up an entry with the current source IP (connect from foreign addr), or current destination (listen on foreign addr). If something is found, the appropriate NAT manip is created using ip_nat_setup_info(). If no appropriate sockrefs are present, the tproxy table is traversed (i.e. proxy requested mappings have precedence over admin rules).

The tproxy table runs after mangle and before nat (priority: -130)

Special UDP Handling

- General assumptions about the proxy:
  - The proxy receives the first packet on a receiver port (not fully specified socket, e.g. remote address 0.0.0.0:0)
  - This initiates a new session
  - The new session opens a new socket (e.g. new local address) and fully specifies this socket by calling connect() with a destination address
  - When another packet is received in the same session, the stack picks the more specific socket (i.e. the session socket)
  - All communication happens in the second packet (races!)

- Translated to netfilter:
  - the first initiator packet should be DNATed to the receiver port, but only once. Further packets will have a seperate conntrack entry (as the DNAT destination will change when the new socket is created)
  - second and further packets will have a real conntrack assigned with TPROXY working similarly as in TCP

Current known usage:

- Zorp, transparent proxy firewall
- patches for squid
- rumors that it works with bridging

Current issues in TPROXY:

- Source port allocation
- UDP support is not complete, though it is enough for Zorp
- timeout updates, or infinit timeouts?

General netfilter additions. These are general, small features that are required by TPROXY and might be useful for other projects as well.

- flags field in struct ip_conntrack for general bitfields
- the flags argument to ip_nat_setup_info(). This is currently used to indicate that no NAT helpers are to be invoked for this NAT mapping.
- sock_release callback to nf_socopts. This is currently used to delete entries from the sockref hashtable when a socket is closed

2.3.1. Current issues with netfilter core

2.3.1.1. Removing Conntrack entries is slow

This would be needed to assure that setting up a new tproxy socket + sockref will result in a successful ip_nat_setup_info() call. Without removing entries address collision will occur and nat setup will fail.

possible solution: In addition to assigning sockrefs to sockets, also assign conntrack entries and remove those as welll when the socket is closed. Problem: locking, reference counting

2.3.1.2. ip_nat_setup_info() hash updating

ip_nat_setup_info() should check nat_info->initialized and update the hashes on it's own. Additionally the check whether the given hook has already initialized a NAT manip could also be moved there.

2.4. Conntrack Failover

Krisztian Kovacs did an experimental implementation of the netfilter failover solution described in the OLS2002 paper by Harald Welte. The implementation should be considered as proof-of-concept implementation. The code has not been released yet, and has not experienced any testing besides a simulated UML environment.

The author and Harald will work together on improving the code and to give it some testing on physical machines. If needed, Astaro would be happy to provide the testlab and pay for travel costs.

2.5. Logging Framework

Jozsef implemented a more general abstraction for a logging interface. This logging interface looks a bit like the current queue handler. Any module can register a logging handler with the core, and everybody who wants to log a packet just calls the logging handler. This way, packets can be logged from outside the LOG/ULOG target - like the TRACE patch or tcp window tracking.

The patch is pending for 2.4 and 2.6 kernel inclusion. It currently has to patch iptables userspace aswell. If the userspace patch can be removed, we would be able to include it without any incompatibilities

2.6. proc patch

- submit proc patch quickly

2.7. raw table

Jozsef Kadlecsik has implemented the 'raw' table. This table registers at PREROUTING with a higher priority than connection tracking. This way a NOTRACK target can be used to exempt certain packets from being tracked by connection tracking. It also implements a TRACE target, that turns on a special bit in the packet. Later in the lifetime of the packet, any subsystem (currently just iptables) can print information about what happens to the packet (certain rule has matched, jumping to different chain, applying a NAT mapping, ...).

This patch is supposed to be submitted immediately (after the logging patch, on which it depends)

2.8. TCP window tracking

TCP windowtracking should be submitted to both 2.4.x and 2.6.x kernels. It should be switched on by default in 2.6.x, _off_ in 2.4.x.

- add tcp rfc793_compatible sysctl, what is default?
- if OOW packet has broken checksum, don't print OOW message (2.6 only)
- Add sysctl to control logging of OOW packets

2.9. nf-hipac

nf-hipac is an excellent packet filter for medium to large sized rulesets. Almost all the functionality of iptables is now supported, without any semantic difference.

The idea is to put the current 2.4.x patch into patch-o-matic. This should give nf-hipac more users. The userspace program remains a the hipac.org website - since they want to know about the approximate number of hipac users.

The 2.6.x version could go in at any time - once it's implementation is finished. However, it is questionable if nf-hipac should be submitted before it's user interface is unified with pkttnetlink.

nf-hipac will not replace {ip,pkt}tables, but be an additional option available to the user. nf-hipac is esp. not suitable for embedded environments, where kernel size and memory usage are very restricted.

2.10. bridgewalling

Bart de Schuymer gave an excellent overview about the current bridging implementation, including ebtables and it's interaction with netfilter/iptables. The details can be found on the slides of Bart's presentation and/or other available documentation on the linux bridge implementation

It is noteworthy that ebtables is not a copy+paste 'port' of iptables, but a reimplementation based on the spirit of iptables. It has several semantic changes/additions, e.g. 'WATCHERS' which are basically a way to have multiple targets in one rule. Those extensions make it difficult to unify ebtables with pkttables. Since pkttables is still mostly vaporware, a discussion on integration with ebtables should be posponed until pkttables works for ip and ipv6.

2.11. Distributions of work within the project

Harald notes that most of the administrative and maintainance work in the project has gradually accumulated as his job. As this is not a problem in itself, it however causes him to be able to spend less time on exciting new

development than he wanted to. The proposal is to offload some of this administrative/maintainance work to other people in order to free up some time.

- website maintainance

  The netfilter/iptables homepage is only maintained at the most minimal level possible. All Harald does is adding items to the 'News' sections and adding new releases. However, there are lots of ideas how the website could be made more attractive. All we lack is somebody actually doing this job.

  - marketing-style information, performance data
  - better link collection, database based
  - personal developer homepages
  - developer diaries
  - FAQ-o-matic system

  As nobody has been volunteering for this job, we will post a call for volunteer

- security incident handling

  From the last couple of security incidents we've learned that we need somebody dedicated for the responsible job of dealing with security incidents. Somebody who can concentrate on this, and to whom this is not just item number 999 on his TODO list.

  - report all security relevant issues
  - coordinate release of advisories with vendors
  - keep advisories on homepage up-to-date

  Oskar Andreasson has volunteered for this job

- mailinglist moderation

  All netfilter lists are set to subscriber-only. This means we will catch lots of SPAM at the administrative interface (since it is just Bcc'ed tothe list). Somebody should reliably check the admin interface at least once per day and take care of moderating the postings.

  As nobody has been volunteering for this job, we will post a call for volunteer

- t-shirt shipping

  The netfilter t-shirts are printed at a printing company in southern germany. It would be very helpful to have somebody volunteering for the work of accepting payment (wire transfer / paypal) and shipping the individual T-Shirts via mail.

Astaro has offered to see if they can somehow handle this

- FAQ maintainance

  Once the new faq-o-matic system is in place, a moderator has to pick the most useful questions from the list of proposed questions, and write or include existing reference answers.

  As nobody has been volunteering for this job, we will post a call for volunteer

  Gert Hansen will work on the faq-o-matic system itself

2.12. test tools

One of the biggest problems during netfilter/iptables development is the lack of test tools. Such tools are important for regression tests after changing the source code, but also necessarry for performance analysis.

2.12.1. regression testing

In the past, the iptables testsuite (cvs 'testsuite' directory) was used to do some minimal regression tests. However, the testsuite has become out of date, and most of the new features of the last two years don't have corresponding tests in the testsuite.

The future of the testsuite was discussed, but according to the authors perception, no clear concencus was established.

2.12.2. benchmarking

In almost any other field of computing, there are standard benchmaks. TPC for databases, htperf for webservers, ... Packet filters do not have such a standard performance test. The closest to a standard performance test is 'number of forwarded packets at a given number of rules without dropping packets'. While this might be suitable to test the performance of a router, it is certainly not suitable for advanced packet filters like stateful firewalls are. Performance is dependent on so many parameters: number of new connections per timeframe, number of state changes within a connection, distribution of l3 and l4 source/destination addresses, ...

In order to test, benchmark and improve performance of the whole linux packet filtering subsystem, we'd need some sophisticated banchmarking utility. Only with this benchmarking utility, we'd be able to compare performance data before and after a code change in a given set of test cases.

Since this problem is not only faced by netfilter developers, but also by vendors like Astaro or Smoothwall - they are both willing to raise some funds for the development of a decent packet filter benchmarking tool.

Since nothing within such a tool would be specific to netfilter, it could be used to benchmark any packet filter / router / ... - and thus to compare performance between different products / projects as well.

2.12.2.1. Harald's connection generator

Harald proposed to write something he calls a 'connection generator'. The idea is to have a set of machines on both ends of the firewall running the connection generator software. The software would have two configuration files.

One file specifies so-called connection profiles. A profile describes the amount of data sent in each direction, the typical duration of the connection, etc. The other file (the test case) specifies the number of connections of each profile, and the addresses between which they should be established: 100 'http' connections, from 192.168.100.0/24, random source to 10.1.2.3:80.

The implementation would be as kernel threads, using the in-kernel sockets API in combination with the TCP zerocopy send path. This way we can avoid implementing our own TCP stack - but are bound to the limitations of the linux stack (in case of scalability due to overhead of sockets api, number of filedescriptors, ...).

2.12.2.2. Rusty's approach to the connection generator

Rusty thinks we definitely need such a test tool (and we should aim it to become the industry standard test tool for firewalls) - but with a different architecture. As opposed to Haralds very primitive design, he thinks one should start the project from the different side: Analyzing real-world connections and classifying them into categories, deriving profiles of common properties of all connections within one class, etc. A connection profile would then resemble the exact timing characteristics of a connection. It would tell us at which sequence number one end was waiting for the reply, when packets had been dropped, how often retransmissions happened, selective acknowledgement was used, ...

The counterpart would then be a replay program that could replay a connection according to a previously-sampled connection. However, it would be able to speed up the connection (by assuming a fixed latency but emulating higher bandwith, ...), and play back tens of thousands of connections at the same time.

The two ends of a connection would always be terminated on the same machine (using two network interfaces). This way we can always be certain about which packets have arrived at what time, and don't need any out-of-band synchronization between sender and receiver. We'd also react realistically to increased latency introduced by the currently tested firewall.

The implementation would not use the existing tcp/ip stack, but rather use a raw socket to gain full control over all timings.

2.12.2.3. Summary

Since there is significant interest in such a project, and even two companies are interested in finacially supporting it, we should continue discussion on the design. At the end of this discussion, a design paper could be published, and a first implementation can be started. We should definitely try to get comments from the academic community, since there might be existing research in this field.

2.13. GPL Violations

- GPL violation - current state - legal action?

2.14. patch-o-matic

The coreteam decided case by case on the future of each patch-o-matic patch. The outcome is summarized by the following table

57_ip_nat-macro-args.patch	apply 2.4.23-pre / 2.6.x
55_ipt_unclean-tcp-flag-table.patch	apply 2.4.23-pre
57_conntrack-tcp-nopickup.patch	remove
HL.patch.ipv6	stay
REJECT.patch.ipv6	needs fix!
fuzzy6.patch.ipv6	combine with nth+random
nth6.patch.ipv6	combine
random6.patch.ipv6	combine
IPV4OPTSSTRIP.patch	stay
NETLINK.patch	stay
NETMAP.patch	submit 2.6
SAME.patch	submit 2.6
TTL.patch	stay
connlimit.patch	needs auditing
iprange.patch	submit 2.6
ipv4options.patch	stay because implementation
mport.patch	stay, too trivial
nth.patch	combine
pool.patch	stay
fuzzy.patch	combine
psd.patch	submit 2.6, needs codingstyle change
quota.patch	stay
random.patch	combine
realm.patch	submit after kaber made indent fix
u32.patch	defer, skb-end/tail bug, ...
condition6.patch	stay
ownercmd6.patch	defer until ipv4 is compatible again
CLASSIFY.patch	submit, MODULE_AUTHOR missing
CONNMARK.patch	submit, CONFIG_IP_NF_CONNTRACK_MARK?
IPMARK.patch	stay
ROUTE.patch	stay
addrtype.patch	defer until kaber fixes
condition.patch	stay
connbytes.patch	stay, replaced by ctstat/netflow
cuseeme-nat.patch	defer, no nat support
h323-conntrack-nat.patch	move to broken
ipt_TARPIT.patch	defer, fix it to use raw table
iptables-loopcheck-speedup.patch	delete!!!
mms-conntrack-nat.patch	stay, no free server
docbook-patch	stay
nfnetlink-ctnetlink-0.11.patch	defer, lockup on SMP!!
owner-socketlookup.patch	submit, ask dave about exported symbol
pptp-conntrack-nat.patch	defer until bugs fixed
quake3-conntrack.patch	apply, no free client+server
rpc-conntrack.patch	stay, not enough users / testers
rsh-conntrack.patch	submit (overwrites name !!!)

string.patch	defer, faster implementation
talk.patch	submit if I find users

3. TODO list for 2.6.0-testX

2.6.0 todo:

- pom rewrite (rusty: python, C)
- remove MIRROR from 2.6 kernel
- remove unclean from 2.6 kernel
- EXPERIMENTAL marks ???
- local nat issues
- add 'all safe netfilter modules' config option
- push all compatibility-breaking stuff to dave
- make ip_queue -> nf_queue (l3 independent)
- submit 03-ipt_REJECT-bridgefix.patch