Summary of the netfilter developer workshop 2004

Harald Welte

et. al.

Summary of the third Netfilter Developer Workshop (Sept 06-07 2004, Erlangen, Germany).

1. List of Attendees

Introductions, around the room

Initials	Name	Affiliation
RR	Rusty Russell	IBM OzLabs
JK	Jeremy Kerr	IBM OzLabs
SS	Stephan Scholz	Astaro
KC	Krisztian Kovacs	Balabit
BS	Balazs Scheidler	Balabit
HN	Henrik Nordstrom	Mara Systems
MB	Michael Bellion	Hipac project
HE	Herve Eychenne	Wallfire project
YK	Yasuyuki Kozakai	Toshiba / USAGI
PM	Patrick McHardy	CoreWorks
MJ	Martin Josefsson	
HW	Harald Welte	hmw-consulting / Astaro
RO	Robert Olsson	SLU University
SH	Stephen Hemminger	OSDL
GH	Gert Hansen	Astaro

2. Introduction

Harald Welte welcomes the 15 attendees of the this third incarnation of the Netfilter Developer Workshop. Thanks go to our Sponsor Astaro, the Linux-Kongress Organizers and University of Erlangen-Nuernberg.

3. Summary of Networking Conference 2004 (Beaverton, OR)

Below is are the raw discussion notes, taken by Rusty Russell:

SH: Main point was that standard firewall reaches dozens to hundreds of rules (multiple interfaces, VPNs, etc). Academic development increasingly on Linux.

HW: Grand Unified Caching: complex setups increasingly in every home, multiple lookups are killing us.

RO: BSD is beating us on raw routing performance. Need better route cache, not a hash lookup.

SH: IPv6 issues, particularly mobile IPv6. James Morris' crypto stuff. 802.11 multiple stacks. UDP fragment. TCP offload issues, driven by "Windows will have it", and RDMA. RDMA-lite? Async-IO for network connections. Arnaldo's cleanup for duplicated code for protocols, should merge. Dynamic window enlarging encouraged window scale to be on, which broke some setups. No performance tests of IPv6. RCU issues under load (solved by call_rcu_bh). Lack of information on enabling different options (queueing discipline, netfilter, iptables, etc). Lack of regression testing of network performance issues. Lack of DoS simulation and measuring handling. skb scatter-gather list change, probably flag day change. Dave has MTLS code, intends to merge. Better route cache.

4. Netfilter relevant work in IETF

4.1. STUN: Simple Traversal of UDP through NAT (RFC3489)

STUN is a protocol typically used for VOIP traversal of NAT. It implements discovery of NAT properties (external, internal addresses) through probe packets. The probe packets are sent back and forth between the client (behind NAT) and some host that is known to have a public IP address.

Based on this discovery, it tries to 'trick' NAT to allocating a particular port binding.

Of course, this only works with certain NAT types (full cone, restricted cone, port restricted cone) - but not with fully symmetric NATs.

netfilter however implements (SNAT and MASQ) as ssymmetric. Thus, STUN does not work with netfilter based NAT.

This is a big problem due to wide use of Linux within access points, gateway products

IETF draft on STUN results with consumer equipment: draft-jennings-midcom-stun-results-01.txt

4.2. IAB Problem Statement about media traffic w/o congestion control (RFC3714)

The IAB fears congestion collapse due to widespread and increasing use of media streaming (VoIP, audio and video streams, ...) without any form of congestion control. Almost all of those services use UDP-based protocols.

IETF starts numerous actions to counter this problem, some of whiche are covered in following sections

4.3. TCP-Friendly Rate Control (TFRC, TFRC-PS)

TFRC is an algorithm, not a protocol. It behaves reasonably fair when competing for bandwith with TCP flows at congested routers. It has a much lower variation of throughput over time and is more suitable for telephony / streaming media content.

TFRC is meant for protocols with a fixed packet size, and sender can moderate number of packets per second.

TFRC-PS (still researched) is meant for protocols with fixed pps rate, but variable payload size.

4.4. Datagram Congestion Control Protocol (DCCP)

DCCP is a datagram oriented transport protocol (layer 4) for unreliable flows. It provides various different cogestion control methods, which can be selected at session startup. At the moment, there is TCP-Like congestion control and TFRC. TFRC-PS is expected to follow.

There is an experimental DCCP Linux 2.4.x and 2.6.x kernel implementation, Arnaldo Carvalho de Melo <acme@conectiva.com.br > is playing with it.

4.5. Next Steps in Signaling Working Group

NSIS develops a protocol called GIMPS (Generic Internet Messaging Protocol for Signalling) that builds on top of TCP/UDP/SCTP/DCCP and can be combined with IPsec and TLS. This protocol is intended for signalling of QoS, NAT and Firewall along the path of a connection.

All messages are sent with Router Alert option in order to be catched by any intermediate Routers/Firewalls/NATs.

There is a number of drafts for the NAT NSIS Signalling Layer Protocol (NAT NSLP): draft-aoun-nsis-nslp-natfw-migration-02, draft-tschofenig-nsis-natfw-security-problems-00, draft-aoun-nsis-nslp-natfw-intrarealm-00, draft-martin-nsis-nslp-natfw-sip-00, draft-fessi-nsis-natfw-threats-01

All in all, it seems incredibly complex and according to Harald very unlikely to be fully implemented over the wide internet.

4.6. BEHAVE Working group

Parts of IETF are aparrently 'acknowledging' presence of NAT, aware of problems, eg. current lack of standardisation.

The BEHAVE Working Group is writing a BCP (best-current-practice) RFC describing the behaviour of a consumer-end NAT device. Probably a second document describing how to design a protocol in order to work over BEHAVE-compliant NAT devices.

It will cover Unicast UDP and TCP, but also Multicast NAT. Apparently some NAT implementations deal correctly with multicast. We should do multicast NAT, too!

The document will most likely encourage (or maybe even require) only outgoing UDP packets to refresh the NAT binding.

problems with source-repeated packets discussion: only a problem where the full tuple is not used RR: may break Real (pnm?) protocol RR: better way of doing timeouts? try a release... ISPs now providing VoIP telephony instead of POTS RR: unaware of linux with broken VoIP implementations? RR: read Kegel recommendation (NAT draft) SH: qdisc detection of congestion control problems Past IETF meeting: congestion mostly occurs at edge links core is always overprovisioned, no packet loss DSL modems, residential distribution this particular setup is likely to lead to congestion collapse

4.7. TODO

DCCP matches, conntrack, NAT

outbound-only refresh

multicast NAT

5. Statue ebtables / bridging

According to Stephen Hemminger, there is not much activity in this area. Recently patches for bridging IPv6 packet filter were posted on netdev.

6. Status patch-o-matic NG

6.1. General discussion about patch-o-matic-NG

First there was a general discussion on the use, purpose and future of patch-o-matig-NG.

According to Harald, users are apparently confued (what should they apply, what makes sense). Also, the categorization of patches (e.g. "WorksForMe") is not very helpful either. He suggested to add a group/category called “kernel track” that would mark all patches that are scheduled to end up in the kernel sooner or later.

Should optimization/cleanup be included in pom-NG, or should go straight into the kernel?

RR: pom was intended for 3rd parties, new extensions originally, use -mm for Netfilter patches

HW: conservative about pushing things into the kernel (bugs)

PM: it’s a bad idea to put a patch first into pom, and then submit

One of the suggestion was to publish a netfilter-test kernel tree that would be the base of all develoment

Also, we should have some system to keep track of the patches, bugfixes. It has to be clearly visible which patch or bugfix went into what kernel release. This also has to include the history of old patches.

Rusty already has something similar (without the history), and Jeremy notes that the PPC64 development team apparently has something similar, too.

BS proposes the use of bugzilla for this. HN notes that there are old bugzilla entries that never get closed. HW adds that bugzilla is easier to search and maintain, even for stuff that is sitting there for a long time.

It is concluded that patches shoud be categorized into bugfix / new feature patches. Bugfix patches are recommended for everybody to apply, new features not neccessarily.

Patch-o-matic is only to be used by new matches, targets, conntrack and NAT helpers. No core infrastructure bugfixes, enhancements or optimizations should go into patch-o-matic.

6.2. Decisions about individual patches

Name	**Comment**	**Merge**
ACCOUNT	per-flow accounting is more important, conntrack_acct is in the mainline kernel, 64 bit counters, available in the proc interface, integrated with ctnetlink; if somebody already uses conntrack gets all these almost for free. Thus, iptables-based accounting is deprecated.	No
CLASSIFY	already in kernel	Already

Name	**Comment**	**Merge**
CONNMARK	should go in, with mark-operations, a similar would be great for mark as well, but that would break the interface (maybe mark2?). compatibility problem: could be able to decide whether the old or new interface is used based on the size of the structure. versioning support should be implemented to get around these limitations. Rusty working on this.	Yes
HOPLIMIT	same as TTL target: no, it's dangerous	No
IPMARK	why is tc not enough to do this? Too obscure	No
IPV4OPTSTRIP	does odd things, may be dangerous	No
MARK-operations	See discussion above. Rusty will work in compatibility mode.	Yes, delayed
NETLINK	Keep it in POM as long as it does not break anything	No
NETMAP	in the mainline now	Already
REJECT	IPv6 reject. Let's push it in, needs some updates	Yes
ROUTE	changes routing, might help on some obscure set-ups	No>
SAME	in kernel, but outdated now, not needed because of NAT core changes anymore. We need to verify it works as expected before removal	Remove
TARPIT	No way	No
TCPLAG	Scientific Use	No
TRACE	broke binary compatibility, so not yet in kernel. Changes iptables structures to be able to give ruleset line number information. HW: let's remove the line number information part, push the other parts in. RR: there are other means of identifying a rule.	Yes, delayed
TTL	Dangerous, same as HOPLIMIT	No
XOR	little practical use, good example	No
account	see as ACCOUNT	No
addrtype	already in	Already
comment	RR: it doesn't hurt if you don't use it, roughly the same as putting the kernel config in the kernel	Yes

Name	**Comment**	**Merge**
condition	Stays in POM, because it's ugly and ruleset updates are faster these days	No
connbytes	connection bytes match, integrated with ct-acct. Push it (we already have the counters). Separate connbytes/connpackets/connavgpkts matches, has a lot of code in common with CONNMARK/state/etc., probably should be unified (binary compatibility problems!) TODO: Harald will unify all of them.	Yes, delayed
conntrack-cacheline-opt		Yes
conntrack-seqfile	already submitted, has problems, RR fixes them	Already
conntrack_arefcount	that's the code which should not be in POM...	?
conntrack_memsave	saves about 40 bytes	Yes
conntrack_nonat	depends on conntrack_locking	?
conntrack_protocol-arrays	RR does not like such "huge" arrays	Yes
ctstat		Already
cuseeme-nat	stays in POM, proprietary protocol	No
directx8-conntrack-nat	evil code and proprietary protocol	No
dropped-table	too obscure	No
dstlimit	supposed to replace static rate-limiting. HW will extend it, rename it to hashlimit.	Yes, delayed
early-drop-norandom	No random early drop needed since our hash distribution is better	Yes
eggdrop-conntrack	Still has no NAT helper	No
expect-optimize		Already
expect-slab-cache		Already
fuzzy	didn't want to unify with nth and random, because it is not statistical mathematics: leave it in POM. RR: What is a FLC?	No
goto	jumps to a chain without returning to the parent chain	No
h323-conntrack-nat	outdated, works only in some cases. TODO: Harald talks to the Gnomemeeting people and checks if it works. Submit if it works.	Yes, delayed
ip_conntrack_count	does not cost anything	Already
ip_queue_nonlinear_skbs	not complete yet	No
ip_queue_vwmark	queue capable of setting packet marks; does not break anything	Yes
ipp2p	like string matching, stays in POM	No

Name	**Comment**	**Merge**
ipt_helper_any		Already
ipt_helper-invert-fix		?
ippool	TODO: ippool vs. ipset vs. everything else (multiple implementation, etc.); development effort should be coordinated (Martin)	No
iptables-loopcheck-speedups	broken	Remove
ipv4options	matches IPv4 header options. TODO: check for PREROUTING, cb may not be available yet	Yes, delayed
mms-conntrack-nat	proprietary.	No
mport	TODO: combine with multiport	No
nat-reservations	stays in POM until we have a helper that needs this code	No
nfnetlink/ctnetlink	Questions: ordered list? unique id effects? generation counter? TODO: use per cpu ids, shift generation counter to the top, add CPU number to the LSBs	Yes, delayed
owner	broken	Already
owner-socketlookup	does not work in PREROUTING ATM, but could be extended	?
owner-supgids		?
policy		Already?
psd		No
quake3-conntrack-nat	propietary	No
quota	does not work on SMP	No
raw		already in
rpc	does not work in some cases	No
rsh	rsh through a firewall? Well, if it works...	Yes
rtsp-conntrack	propietary (RR: is there now a FOSS server?)	No
talk-conntrack-nat	Jozsef's decision	Yes
tcp-window-tracking	TODO: benchmark between the old and the new code (2.6 before, 2.6 after the patch was applied)	Already
time		No
tproxy	no GPL headers,should export _confirm as GPL only symbol, #define cleanups, should go in	Yes, delayed
xover	it was a quick hack	Remove

TODO: fix the logic to not try to apply updates for old kernels make sure only updates go to the updates

directory remove all the patches modifying core parts from POM

7. Status of ct_sync

ct_sync is the ip_conntrack state synchronization mechanism for stateful failover.

About 1 year ago there was Krisztian's prototype implementation. Harald took over, cleaned it up and added important features (l2drop, internal notrack hook, full resync) but it still contained a lot of bugs.

After a couple of weeks work by Krisztian, we now have a working ct_sync implementation.

The only major missing part: is expectation support. Especially difficult is the order of full synchronization (master - expect - slave connection).

There is also the expectfn() problem: Currently the expectfn is a pointer, which is difficult to replicate. Jozsef's idea: move the function pointer to ip_conntrack_helper and register dummy helpers in case there is a expectfn() without helper.

In it's latest state, the ct_sync code only replicates connections that are explicitly marked by connmark.

Following discussion:

RR: do not replicate locally terminated connections, because there's no point socket state)

HW: by default we synchronize everything, just like in conntrack

HN: default for local is not to replicate, everything else replicated; but ability to change this default

HW: this may easily cause clashes in the NAT table because of null binding; there are not too much local connections anyway. Are tproxy connections local?

KK: yes, they are, proxy state and proxy connections are not replicated anyway.

RR: but we want to get an RST or ICMP port unreachable back.

Decision: replicate everything by default, and rely on the admin make exemptions

Protocol problems: Recovery: All packets are sent to all nodes (multicast). This can be optimized by using unicast messages for recovery, and multicast for normal sync. Doesn't need any protocol changes.

KK suggests resending is not needed at all, because it complicates things and happens in rare cases

HW: We need to deploy it first and then gather some statistics.

Update messages:

KK: update messages are too long (240 bytes each

HW: the goal was to reduce PPS and not bandwidth

KK: TCP window tracking replication would need shorter messages

HW: TCP window tracking should revert to sloppy mode after a failover

HW: reducing the message size does not really help, because we'd loose more synchronization messages in case of a failover (more messages are being accumulated to be sent)

agreed: use sloppy mode and wait until somebody complains

Future plans: Currently ct_sync is strictly for Linux 2.4, port to Linux 2.6 (IPv4 only) Linux 2.6 port is not expected to be difficult (but it's not there yet) In the long term, a nf-conntrack port is required.

Integration:

HW: ctnetlink events are needed for integration (...). However, ctnetlink affects performance. It would be a good idea to split ctnetlink notifiers from the real netlink messaging (which is not needed by ct_sync)

8. Status of nf_conntrack

A first port of ip_conntrack to IPv6, called ip6_conntrack, was implemented 2 years ago by Toshiba Corp. This port was rejected, since we already have way enough replication with ip_tables/ip6_tables, don't want to make the same mistake again.

The solution is the proposed (HW) and successfully implemented (YK) nf_conntrack. It generalizes the abstraction of a connection, and modularizes the layer 3 protocol specific parts. It only requires very small changes to existing layer 4 protocol modules or application helpers.

However, the conntrack tuple is a union of ipv4 and ipv6 specific parts, and thus enlarged because of 128bit addresses.

In order to counter this enlargement of struct nf_conntrak, we now implemented variable-sized nf_conntrack's, depending on the features a specific connection needs. If there are no helpers, 152 bytes can be saved by not allocating the helper-specific part. A number of slab caches is created for the different sized structures, and a bitmask in the static header indicates the size of the remainder.

One remaining difficulty is the issue of fragmented IPV6 packets

IPv6 doesn't know the concept of router-based fragmentation. Therefore, the IPVv6 stack will send "packet too big" errors if we hand it oversized (defragmented) packets. Therefore, nf_conntrack needs to defragment, but still keep the original fragments.

Solution is to defragment into a shadowed clone only used by conntrack/ip6tables. The rest of the stack sees only the

Since IPv6 fragmentation is specified as header option, it is therefore currently implemented in a protocol helper module. This is not the intended behaviour, especially since it stores per-skb information in struct nf_conntrack, which is broken. HW will investigage the issue (he has already published his results at the time the proceedings were written) and YK will change implementation.

Issues in Size optimization, slab caches of needed structure sizes: 2^n possible depending on the combinations of conntrack fields needed, type of helpers etc. This is not considred a real problem, since 'n' isn't large at the moment (2).

When to destroy the needed slab caches when unloading helper modules? Many problems discussed, but is this even needed? Suggestion from the workshop (Harald and Rusty) is to ignore the issue, just keep the ever referenced slab cache sizes around until the whole of nf_conntrack is unloaded. Conclusion: Not a problem

TODO: fix userspace including ip_conntrack.h. This is a kernel-only header.

synchronize with recent ip_conntrack. Currently up to 2.6.8.1 but ip_conntrack is a moving target...

HW: NAT (IPv4). With a design criteria of not beeing easy to apply to IPv6.

NAT-PT (IPv4<->IPV6). Does this need to touch nf_conntrack at all? Maybe it is good, maybe this is better done as a transform?

9. non-linear skb packet pattern matching

Why do we need this? conntrack helpers/NAT need pattern matching in payload e.g. string, l7filter, ... This is hard to implement with non-linear skb's, and all of the current implementations are ver inefficient (linearize skb, do unoptimized dumb string search).

The goal is to let helpers call other kernel part for pattern matching, so netfilter does not have to worry about an efficient implementation of pattern matching.

There alread is libqsearch from Phillipe Biondi. It implements an abstract interface for pattern matching. Algorithms are plugins, and it can be compiled as userspace (.so) or kernel code (.ko). Different algorithms available: string, wildcards, regular expressions, .. However, this code can not be used without modifications (semantic of calls, code not compatible to kernel coding style, ...)

HW is currently re-working the libqsearch code to have caller-allocated structures (as opposed to callee-allocated) and to comply with the kernel coding style. This includes integration with skb_iter() by RR in order to support nonlinear skb's

10. nfnetlink, ctnetlink, pkttnetlink, pkttables, libpkttables

Harald came up with a Version-TLV (VTLV) based aproach, this came out of the pkttables requriement from last years nf workshop.

Every VTLV consists of an array of 256 callback functions (are they enough?) The problem is that if there are to many redundant information in every TLV.

We also need to keep in mind, that dumping the complete conntrack table for example, generates quite an amount of data that need to able to handled properly.

We need the portablity of pkt tables to easily run on different Architectures. Is the effort of going the unified VTlV way worth it? Do we really want to have TLV's for every single data element? Or are there any alternatives? A template based TLV system is proposed, where the initial message specifies the complete structure. following messages will follow this initial message. This makes sense for fixed-size entries such as conntrack / expect, but not so much for pkt_tables rules.

Why do we need TLV? TLV would be needed to be easy extensible in the future.

netlink2 was designed as a endian independed system to transfer control information between different sub/systems. netlink2 is now an officilal IETF RFC. nfnetlink is targeted to run on top of netlink2.

Jamal is working on implementing this for the routing code, status unclear.

The question is how the big the performance degredation is due to this? Harald assums about twice the effort.

Plugin Architecture of pkttables Harald changed the new plugin architecture, rather every module parses its options themselves, they now specify their options and parsing is done by the core libpkttables.

As a result of doing this, a module just consists of big datastructure and the registation function, therefore this could also be stored in some textual description, such as a XML-style configuration datastucture file.

HN mentioned that this does not work for all matches like CONNMARK, which have quite complicated option syntax. HW was not aware of this, and currently doesn't have a proposal how to integrate this.

HW thinks that this should be added to the core parser as new datatypes, rather than having the parser in the plugin modules. But this would prevent people from easily extending it. A possible fix would be parser plugins to the core, that can by dynamically changed and used by different modules.

TODO: HW needs to look into all matches again, to assure they can somehow be ported to pkttables

Conclusion: Currently we'll stick with the C-plugin style for now, txt/xml style should be reconsidered next verision.

Error handling. PM mentioned that error checking and argument handling should not be part of the kernel, but be checked in userspace. The big question is, if ruleset validation should be done in userspace, in the kernel or both? The general conclusion of the discussion is that the kernel should only do very minimal checking and trust the information from userspace.

It is the declared goal of pkttables to serve as configuration frontend for other packet matching engines such as nf-hipac. Coordination is needed.

HW announced that at some point he will take "pkttables holidays", that is he'll disappear from normal netfilter/iptables development and concentrate on pkttables for two weeks. Hopefully this will speed up progress in this area.

11. Status of nf-hipac

Michael Bellion is the only developer left working on nf-hipac, since Thomas heinz cannot afford spending any more time on th project.

Existing released GPL licensed code (for 2.4.x) has known bugs, and the algorithmic core some shortcomings; mainly memory usage

Plans for a reimplementation of algorithmic core (under 2.6.x, with RCU) exist for quite some time, as have considerations to make this code closed source in order to earn money.

nf-hipac has now succeeded in finding a sponsor for this new implementation: Mara Systems A.B. This will ensure nf-hipac will stay GPL.

Timeframe for new implementation is about one year from now.

Kernel-/Userspace Interface has to be shared with pkttables. Harald will make sure the respective parts exist beforehand.

12. nf-sim: The netfilter simulator

Jeremy Kerr has been working on nf-sim, a userspace netfilter simulator. Project is available from http://ozlabs.org/.

nf-sim is to be used for regression tests, as replacement for the bit-rotten testsuite in netfilter CVS.

nf-sim works 100% in userspace. It grabs code from any existing 2.4.x / 2.6.x kernel tree, and compiles it into userspace binary. It thus emulates the minimal subset of kernel functions in order to make the netfilter code running. LD_LIBRARY_PRELOAD wrappers exist in order to redirect iptables' {set,get}sockopt calls into the simulator. Time is not running, but 100% controlled by user. Any interaction with the simulator happens on an interactive commandline shell (supports tab-completion, commandline editing and history).

Currently it is not possible to inject packets from external sources, such as ethertap / pcap-savefiles / ...

13. profiling/benchmarking

13.1. profiling/benchmarks by RO

RO is doing nigh-performance routing/forwarding benchmarks with linux-based routers for many years. Since his university couldn't afford any packet generators for testing, he has developed his own packet generator, initally called tg3, now called pktgen.

For a more detailed description of pktgen, pleae refer to his pktgen paper, as published in the Linux Kongress 2004 proceedings.

He is currently working on a new version, that includes multiple flow support and ipv6 packet generation

13.2. Some forwarding-only Performance Data (RO)

Increase from P3 to Xeon did not match clock rate 350kpps to 500kpps per outbound e1000 interface.

ip stack needs help to get better SMP scaleability

with perfect affinity CPU's get only 20% gain with second Xeon

Opteron's (1.6Ghz) on 1UP 850kpps. Get much better CPU scaleability 80/90% with second CPU.

Apparently at 850-860kpps there is some hardware limit because of the PCI arbitration latency that is neccessarry for every TX DMA. This is supposed to become better with PCI Express, but RO hasn't had any hardware yet.

IP forwarding might be a good fit to NUMA, need memory allocation with CPU localization

13.3. Netfilter benchmarkign by HW

Problems with Intel 4 port cards because of additional PCI bridge. Lose 30% of performance (850kpps to 500kpps)

PCI-X will have similar problems with multi-port cards E1000 PCI-x is broken with message signal interrupts.

With forwarding, if you affinity interrupt to processor then each skb ends up being freed on different processor. If boards supported separate tx/rx interrupts then this would fix the problem. However, there is only a single 10Gbit board (no Gbit boards) from S2IO that support this feature (stupid hardware designers).

Other possibility would be for driver to attempt cleanup of transmit skb's in transmit routine (but this is very opportunistic).

Initial rate (forwarding only) 800kpps

insmod ip_conntrack -200 kpps

load IPtable (even empty) 25%

oprofile (non-halted) everything in ip_tables (3%)

static compiling makes 5% difference

full test (nat, mangle, filter, ip_conntrack): down to 350kpps

building as 64bit kernel hurts performance, this is assumed because skb grows another cache line because of large pointers

14. netfilter e.V.

HW proposes to found a legal body for the netfilter project. This can be done as charitable organization under german law (so-called "gemeinnuetziger eingetragener Verein").

It's purpose would be to ease relationship with commercial vendors of netfilter/iptables based products. They are used to deal with legal entities, such as consortia, where they pay membership fees, and have some representative communicate their needs/requiremetns towards this organization.

This would also make it way easier for commercial entities to make donations to the projects... For tax reasones, it's way easier to make a donation to a legal entity than to some individual developer.

Harald is working on the bylaws of such an organisation at the moment. There are lots of detail questions, which he doesn't want to discuss at the workshop at large.

However, one fundamental issues is that somebody has to do the work. Somebody has to contact the vendors, keep communicating with them, etc. HW wouldn't mind doing this work, but he already has lots of time

constraints. Unfortunately nobody at the workshop volunteered to do this. HW will try to find a volunteer for this, maybe he can convince somebody with PR background.

Despite the lack of volunteers, HW will pursue founding the organization. Also, the costs associated with founding/registering are quite low.

15. Linux 2.6.x, netfilter and IPsec

The Linux 2.6.x IPsec implementation doesn't really match well with netfilter/iptables. At the moment, especially connection tracking and NAT will break in many of the common usage scenarios of the IPsec stack.

PM was working for quite some time on a series of patches adressing this problem. None of them have made it into mainline yet, unfortunately.

15.1. locally-originated encapsulated packets

For locally-encapsulated traffic, we see the plain traffic in NF_IP_LOCAL_OUT, and the encrypted traffic in NF_IP_POST_ROUTING. However, to make all of netfilter work, we need plain traffic in NF_IP_LOCAL_OUT, NF_IP_POST_ROUTING, and then the encrypted traffic in NF_IP_LOCAL_OUT and NF_IP_POST_ROUTING.

The patch replaces dst_output() calls in net/ipv4/ by ip_dst_output(), which calls NF_HOOK(NF_IP_POST_ROUTING) if dst->child != NULL. This means we see the plain paket in POST_ROUTING before encryption.

xfrms mark packet as transformed in the IPCB. ip_output() is renamed to ip_output2(), and a new ip_output() function calls NF_HOOK(NF_IP_LOCAL_OUT). This makes sure that encrypted packet hit LOCAL_OUT before POST_ROUTING.

15.2. forwarded and decapsulated packets

No ip_route_output() call, but xfrm_lookup() is called from ip_forward(). The only difference as far as netfilter is concerned: Forwarded packets are handled exactly as outgoing packet. The hook-order is: plaintext->FORWARD->POST_ROUTING->encrypted->LOCAL_OUT->POST_ROUTING.

15.3. decapsulated locally-terminated traffic

Incoming encrypted packets go up the stack, traversing the hooks as usual. ipsec layer4 protocol handler loops until all transport mode SAs have been handled. If next SA is tunnel mode, packet is reinjected to the stack via netif_rx(). Otherwise, there is no further SA, pcaket is handled via ip_local_deliver_finish().

Problems with netfilter: Packets delivered to the final protocol handler don't traverse netfilter hooks after decryption. Packets reinjected to the stack will traverse hooks again. Intermediate processing steps shouldn't be visible to the user.

Current solution: Packets are marked (skb->sp->decap_done == 1) and reinjected into the stack once we know decryption is done. Decryption is done if the packet was reinjected into the stack by ipsec and dst is non-local after routing, or the protocol handler is not marked with new flag xfrm_prot(ip_local_deliver_finish). Packets in intermediate processing steps are hidden from the hooks (skb->sp != NULL), only final reinjected packets is visible (skb->sp != NULL && skb->sp->decap_done)

Lots of problems: In tunnel mode, packet will go through the stack in decrypted state twice. Netfilter only sees them once, but stack twice. Besides overhead, not nice for statistics.

IPCOMP packets may be sent as IPIP when compression is ineffective, but IPIP is no ipsec protocol. So decryption is also done if packet in ipip.c:ipip_rcv() is for a local tunnel and packet came from ipsec (skb->sp != NULL && !skb->sp->decap_done).

Same for invalid AH/ESP/IPCOMP packets, they are invisible to netfilter if the respective protocol is registered.

15.4. NAT and output packets

xfrm_lookup() is done with routing. NAT in PRE_ROUTING doesn't pose any problem, NAT in POST_ROUTING: We need to do xfrm_lookup() again.

The current patch adds policy lookups to ip_route_me_harder() and reroutes in POT_ROUTING if the key used for looking up the policy has changed.

This has unexpected effects on source-based policy routing.

15.5. NAT and input packets

__xfrm_policy_check() checks if the struct flowi derived from the packet matches the selectors of used SAs. However, the NATed packet doesn't match anymore.

Solution: Add nf_nat_decode_session() to make struct flowi look like before NAT

15.6. Matching decapsulated IPsec packets

People want to match if a packet was decapsulated by ipsec. This used to be simple with freeswasn because of the virtual ipsec interfaces.

New policy match allows to match policy that will be used for outgoing packets and SA's that were used during decapsulation for incoming packets.

16. GPL Enforcement Status

HW gives an overview about his past and present GPL enforcement efforts. Most details of the settlements will not be disclosed in this proceedings document, just a quick overview.

So far 10 out-of-court settlements (amicable agreements) with companies such as Allnet, Siemens, Fujitsu-Siemens, Securepoint, Gateprotect, Asus, Belkin, ...

Only one case had to go to court: Sitecom. We got a preliminary injunction. Sitecom appealed, Sitecom lost the appeals case. Sitecom did not appeal again, and the deadline for another appeal has already passed. Meanwhile, they have been in violation of the injunction, and we applied at the court to fine them for breach of a court-ordered preliminary injunction.

HW is working on more cases, since apparently there are more and more products basing on netfilter/iptables without adhering to the license conditions