An Introduction to Xtables-addons

Jan Engelhardt

Presented at NFWS 2008

2008-09-30

Table of Contents

patch-o-matic (Aug 2002–2003) and p.o.m.-ng (Nov 2003–2007)

- package to hold extensions not merged yet in mainline

patch-o-matic (Aug 2002–2003) and p.o.m.-ng (Nov 2003–2007)

- package to hold extensions not merged yet in mainline and those that would never go in anyway.

patch-o-matic (Aug 2002–2003) and p.o.m.-ng (Nov 2003–2007)

- package to hold extensions not merged yet in mainline and those that would never go in anyway.
- development playground – convenient patch scripts (at that time)

Pitfalls

- people patched lots of it in
  Despite the warning

  > *"Each patch is a new feature: many have minimal impact, some do not. Almost every one has bugs, so I don't recommend applying them all!"*

  - in retrospect, they certainly had bugs

Pitfalls

- people patched lots of it in
  Despite the warning

  > *"Each patch is a new feature: many have minimal impact, some do not. Almost every one has bugs, so I don't recommend applying them all!"*

  - in retrospect, they certainly had bugs
- distributions (Debian, PLD Linux, OpenWRT) patched a few features in sometimes
  - maintenance cost of carrying and updating the patches
  - usually split over two packages (kernel, iptables)

Patching

- can create merging conflicts when patches are applied
  - patch not updated for most recent kernel
  - patches can conflict among themselves

Patching

- can create merging conflicts when patches are applied
  - patch not updated for most recent kernel
  - patches can conflict among themselves
- possibility of incorrect conflict resolution by a novice user

Patching

- can create merging conflicts when patches are applied
  - patch not updated for most recent kernel
  - patches can conflict among themselves
- possibility of incorrect conflict resolution by a novice user
- the patch might even apply cleanly

but the resulting source code may still have flaws.

What could be wrong here?

A sample match function that never matches

```
static int throw_away_match(const struct sk_buff *skb,
    const struct net_device *in, const struct net_device *out,
    const struct xt_match *match, const void *matchinfo, int offset,
    unsigned int protoff, int *hotdrop)
{
        if (uncorrectable_error)
                *hotdrop = 1;
        return 0;
}
```

What could be wrong here?

A sample match function that never matches

```
static int throw_away_match(const struct sk_buff *skb,
    const struct net_device *in, const struct net_device *out,
    const struct xt_match *match, const void *matchinfo, int offset,
    unsigned int protoff, int *hotdrop)
{
        if (uncorrectable_error)
                *hotdrop = 1;
        return 0;
}
```

ABI/API mismatch

Newer kernels require `bool *`. Dereferencing `hotdrop` here causes a write of 4 bytes into a memory region that is just 1 byte usually.

Code quality

There's more! Code was often plagued with various issues – though this is a result of the particular developer, not POM.

- variable-width types
- unaligned access
- endian correctness
- running sparse is advised (`make C=1`), as is review

Size mismatch

Types with variable width across different arches

```
struct ipt_ipmark_target_info {
        unsigned long andmask, ormask;
        char addr;
};
```

- will fail in mixed-bitness environments (commonly done on sparc64) unless additional compat code is present
- often went unnoticed because most people used x86 32-bit installs

Kernel message

```
x_tables:  connmark match:  invalid size 24 != 12
```

Solution

Only use `char`, `__u8/16/32/64` (and signed variants) types and structs/unions/arrays of these. For exceptions, see the Documentation.

Alignment violation

Unaligned access

```
#define get_u16(X, O) (*(const __u16 *)((X) + O))

if (get_u32(payload, 33) == __constant_htonl(0x71182b1a) &&
    get_u16(payload, 147) == __constant_htonl(0xf792)) {
        printk(KERN_INFO "got WinMX\n");
        return IPP2P_WINMX * 100 + 4;
}
```

- often goes unnoticed because x86 handles it transparently

Resulting oops on sparc64

```
Kernel unaligned access at TPC[79c344] search_winmx+0x123/0x789
```

Safe version

```
#define get_u16(X, O) get_unaligned((const __u16 *)((X) + O))
```

- wrong argument types, wrong number of arguments or order of these in a call/function head

- wrong argument types, wrong number of arguments or order of these in a call/function head
- resulting compiler warnings ignored by the novice user (“it compiles? ship it!”)

- wrong argument types, wrong number of arguments or order of these in a call/function head
- resulting compiler warnings ignored by the novice user (“it compiles? ship it!”)
- silent corruption, kernel oops and an unhappy user.

```
\|/ ____ \|/
"@'/ .. \`@"
/_| \__/ |_\
   \__U_/
(sparc64)
```

```
^__^
(xx)\_______
(__)\       )\/\
 U  ||----w |
    ||     ||
(parisc)
```

Maintenance

- extensions had to be updated whenever the kernel API changed

Maintenance

- extensions had to be updated whenever the kernel API changed *every single* extension – more than 30 during prime time

Maintenance

- extensions had to be updated whenever the kernel API changed *every single* extension – more than 30 during prime time
- frowned-upon `#if` forest to make code work across APIs of multiple versions
- workarounds replicated among all extensions –
  that is, if they were updated at all

#if forest example

```
#if LINUX_VERSION_CODE >= KERNEL_VERSION(2,6,23)
static bool ipt_acc_checkentry(const char *tablename,
#else
static int ipt_acc_checkentry(const char *tablename,
#endif
#if LINUX_VERSION_CODE >= KERNEL_VERSION(2,6,16)
                              const void *e,
#else
                              const struct ipt_entry *e,
#endif
#if LINUX_VERSION_CODE >= KERNEL_VERSION(2,6,17)
                              const struct xt_target *target,
#endif
                              void *targinfo,
#if LINUX_VERSION_CODE < KERNEL_VERSION(2,6,19)
                              unsigned int targinfosize,
#endif
                              unsigned int hook_mask)
```

Conclusion

- code updates do not scale
- patching the kernel source may incur traps

Conclusion

- code updates do not scale
- patching the kernel source may incur traps
- recompiling the kernel takes its time
- voids the automatic stable/security updates your distro provides

- a lot of extensions got marked as deleted in the VCS (May 2006)
  - some were merged since 2.6.14 (Oct 2005) already

    mport/multiport, iprange, NETMAP, comment, goto, NETLINK/NFQUEUE, unclean(partial)
  - FYI: Linux 2.6.17 released in June 2006
- user demand for non-standard extensions still there

- a lot of extensions got marked as deleted in the VCS (May 2006)
  - some were merged since 2.6.14 (Oct 2005) already

    mport/multiport, iprange, NETMAP, comment, goto, NETLINK/NFQUEUE, unclean(partial)
  - FYI: Linux 2.6.17 released in June 2006
- user demand for non-standard extensions still there
- more extensions found their way into mainline later
  - 2006: nth, quota, random
  - 2007: TRACE, connrate/rateest, connlimit, time, u32
- other extensions have gone into Xtables-addons
  - 2008: IPMARK, TARPIT, condition, fuzzy, geoip, ipp2p
  - 2009: ipv4options

- About 8 or so “left” in the depths of the POM history.
  No real demand for these.
- (Update: Remaining code deleted December 2008)

What it is

"*Xtables-addons is the successor to patch-o-matic(-ng). Likewise, it contains extensions that were not accepted in the main iptables package [so far]*".

Same idea, different implementation.

What it is

"*Xtables-addons is the successor to patch-o-matic(-ng). Likewise, it contains extensions that were not accepted in the main iptables package [so far]*".

Same idea, different implementation.

`http://xtables-addons.sf.net/` (homepage)
`git://xtables-addons.git.sf.net/gitroot/xtables-addons/xtables-addons` (clone)
`http://xtables-addons.git.sf.net/` (gitweb)

How it works

- no patches (`.diff` files) or POM trees
- plain source code and Makefiles
- only requires the kernel build environment, full source not needed (`/lib/modules/$version/build/`)

How it works

- no patches (`.diff` files) or POM trees
- plain source code and Makefiles
- only requires the kernel build environment, full source not needed (`/lib/modules/$version/build/`)
- extensions built as modules
- no reboot, instant use – also perfect for development
- works with the distro-provided kernel
  (i. e. not having to roll your own and miss out on distro kernel updates.)

How it works

- no patches (`.diff` files) or POM trees
- plain source code and Makefiles
- only requires the kernel build environment, full source not needed (`/lib/modules/$version/build/`)
- extensions built as modules
- no reboot, instant use – also perfect for development
- works with the distro-provided kernel
  (i. e. not having to roll your own and miss out on distro kernel updates.)
  - kernel 2.6.17 or up (2¼ years old as of Oct 2008, so good coverage)
  - minus points for distros doing excessive backports (CentOS5) – one needs to hand-tweak the Xt-a source and remove what has already been backported.

Implementation

- uses an extra API layer so that extensions remain relatively clean of version-related `#ifs`.
  `grep '^#if LINUX_VERSION' xt_*.c`
  8 (for 15 extensions, 0.53/file)
- uses glue functions and macro-based redirection (`compat_xtables.[ch]`)
- most extensions need no more than `#include "compat_xtables.h"` as the last include directive

Limitations

- patching the kernel source, like header files (as ACCOUNT, IMQ and layer7 require), is not within scope.
    - but you could still make use of the glue code for the parts that do not patch existing files
- compiling extensions into non-modular kernels seems possible, but no demand so far

  (`cd linux/; ln -s ../xtables-addons;` and edit some kernel Makefile to descend into `xtables-addons/extensions/`)

Current state

Problems with code resolved when it was imported into Xtables-addons.

- works in mixed-bitness environments
  e. g. 64-bit kernel and 32-bit userspace
- (believed to be) alignment- and endianess-correct
  (unfortunate lack of non-x86 hardware to fully test)
- added IPv6 support to some extensions

Extensions

(By various authors.)

- condition – match on a flag changable from userspace, e. g. (discrete) weather condition (see other talk).

Extensions

(By various authors.)

- condition – match on a flag changable from userspace, e. g. (discrete) weather condition (see other talk).
- geoip – match on countries.

  "Microsoft and Google I can live without, but an internet without North Korea is no internet I want to be a part of!"

  —leoc on LWN.net

```
-A INPUT -m geoip --src-cc KP -j ACCEPT
```

Extensions

(By various authors.)

- condition – match on a flag changable from userspace, e. g. (discrete) weather condition (see other talk).
- geoip – match on countries.

  "Microsoft and Google I can live without, but an internet without North Korea is no internet I want to be a part of!"

  —leoc on LWN.net

  ```
  -A INPUT -m geoip --src-cc KP -j ACCEPT
  ```
- TEE – reroute a copy of the packet

Extensions

- TARPIT – hold TCP connection indefinitely. Use this for port 25 if you do not run a receiving server.

Extensions

- TARPIT – hold TCP connection indefinitely. Use this for port 25 if you do not run a receiving server.
- DELUDE – does TCP handshake, but close connections afterwards. Thwarts nmap stealth scans. (Also see CHAOS for combined portscanner countermeasures.)

Extensions

- TARPIT – hold TCP connection indefinitely. Use this for port 25 if you do not run a receiving server.
- DELUDE – does TCP handshake, but close connections afterwards. Thwarts nmap stealth scans. (Also see CHAOS for combined portscanner countermeasures.)
- (more added over time)
- Sample modules for documentation

Availability

- As of 2009: Alpine Linux, CRUX, Debian, Gentoo, OpenWRT, Polish Linux Distribution (PLD), openSUSE, Shorewall, Slackware.
- presenter (that's me) has RPMs for openSUSE
- `http://freecode.com/projects/xtables-addons/`

Documentation

- “Writing Netfilter modules”
  Book in PDF format on `http://inai.de/`