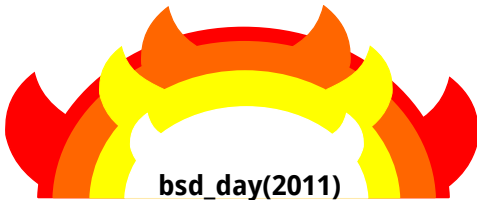


NPF: a new packet filter

Zoltán Arnold Nagy
zoltan@NetBSD.org



Slovak University of Technology,
Bratislava, Slovakia
November 5, 2011

- ▶ What is a packet filter?



- ▶ What is a packet filter?
 - ▶ A boolean valued function ($f : \mathbb{P} \rightarrow \mathbb{B}$)



- ▶ What is a packet filter?
 - ▶ A boolean valued function ($f : \mathbb{P} \rightarrow \mathbb{B}$)
- ▶ How can it be implemented?



- ▶ What is a packet filter?
 - ▶ A boolean valued function ($f : \mathbb{P} \rightarrow \mathbb{B}$)
- ▶ How can it be implemented?
 - ▶ boolean expression tree
 - ▶ directed acyclic control flow graph
 - ▶ if-then-else hell



A sea of firewalls

- ▶ IPFilter (ipf)
- ▶ FreeBSD's ipfw
- ▶ OpenBSD's pf
- ▶ NetBSD's npf



How NPF started out

- ▶ Sponsored by The NetBSD Foundation
- ▶ Written by Mindaugas Rasiukevicius (rmind@) from scratch, although the design was inspired by the Berkeley Packet Filter
- ▶ First imported to -current in August 2010
- ▶ Will be widely available with the 6.0 release
- ▶ First step in improving NetBSD's networking capabilities
- ▶ Second step: removing the big kernel lock (proposal deadline: Oct 31st)



Google Summer of Code

- ▶ Goal: The program's goal is to promote open-source software development among students and get them actually involved



Google Summer of Code

- ▶ Goal: The program's goal is to promote open-source software development among students and get them actually involved
- ▶ Mentoring organizations apply with a set of proposed projects



Google Summer of Code

- ▶ Goal: The program's goal is to promote open-source software development among students and get them actually involved
- ▶ Mentoring organizations apply with a set of proposed projects
- ▶ Students apply for organizations with a proposal based either on a proposed project or a new one



Google Summer of Code

- ▶ Goal: The program's goal is to promote open-source software development among students and get them actually involved
- ▶ Mentoring organizations apply with a set of proposed projects
- ▶ Students apply for organizations with a proposal based either on a proposed project or a new one
- ▶ Organizations get a number of slots for students



Google Summer of Code

- ▶ Goal: The program's goal is to promote open-source software development among students and get them actually involved
- ▶ Mentoring organizations apply with a set of proposed projects
- ▶ Students apply for organizations with a proposal based either on a proposed project or a new one
- ▶ Organizations get a number of slots for students
- ▶ Mentoring organizations get \$500 per student, students get \$5000



Google Summer of Code

- ▶ Goal: The program's goal is to promote open-source software development among students and get them actually involved
- ▶ Mentoring organizations apply with a set of proposed projects
- ▶ Students apply for organizations with a proposal based either on a proposed project or a new one
- ▶ Organizations get a number of slots for students
- ▶ Mentoring organizations get \$500 per student, students get \$5000
- ▶ It has been running since 2005
 - ▶ 2011: 175 mentoring organizations, 1115 students
- ▶ NetBSD has participated every year so far with a high success rate
 - ▶ 2011: 9 projects, 8 projects ended with success
- ▶ My GSoC proposal for 2011 was to add IPv6 support to NPF
- ▶ ...was finally committed yesterday around 2 AM :)



Motivations and goals

- ▶ There are a few existing firewalls
- ▶ It's easier to design a new firewall from ground up than to clean up existing codebases
- ▶ Design goals for NPF:
 - ▶ MP-safety and locklessness for scalable MP performance
 - ▶ Fast tree- and hash-based lookup support for tables
 - ▶ Stateful packet filtering
 - ▶ N-Code processor, a general bytecode engine
 - ▶ Keep configuration syntax changes to a minimum
 - ▶ Modularity, extensibility: an extension API for developers, hooking support
 - ▶ Last but not least: simplicity
- ▶ Of course it's portable, uses pfil(9) hooks; DragonFlyBSD is considering adoption
- ▶ Please keep in mind: it's a moving target currently



Usage

- ▶ `npfctl` can be used to communicate with `/dev/npf` via `ioctl`s
 - ▶ start, stop, reload, flush, stats, tables, sessions, ...
- ▶ Rules will be compiled to a bytecode, and it will be loaded as `ruleset(s)`
- ▶ Current parser will be replaced by a new one by Martin



What can it do today?

- ▶ Rule syntax is nearly identical to other firewalls
- ▶ Group support
- ▶ Rule procedures (connection-based packet transformations)
 - ▶ IP ID randomization
 - ▶ enforcement of TCP minimum TTL
 - ▶ enforcement of TCP Maximum Segment Size (MSS)
 - ▶ logging
- ▶ Tables support
- ▶ Application-level gateways (ALGs)




```
int_if = "wm0"
ext_if = "wm1"
table "1" type "tree" dynamic
procedure "rid" { normalize (random-id) }
procedure "log" { log npflog0 }
group (name "external", interface $ext_if) {
    block in quick from <1>
    pass out quick from $ext_if keep state apply "rid"
    pass in quick proto tcp to $ext_if port ssh apply "log"
    ...
}
group (name "internal", interface $int_if) {
    block in all
    pass in quick from <1>
    pass out quick all
}
group (default) { block all }
```



Inside

N-code engine:

- ▶ General purpose bytecode engine, 32-bit words, 4 registers available
- ▶ The firewall configuration is compiled to our bytecode format, then loaded
- ▶ CISC and RISC-like instructions
- ▶ The packets are processed as a byte-stream

Efficient internal structures

- ▶ `npf_addr` (`in6_addr`, 128-bit) for addresses (the first 32 bit is used for IPv4 addresses)
- ▶ `uint_fast8_t` for masks
 - ▶ instead of generating the appropriate `npf_addr` value from 255.255.255.0 or /24, we just store the mask as a number
 - ▶ tradeoff: CPU for memory



Inside

Let's see what we can work with...

```
typedef struct {
    /* Information flags. */
    uint32_t      npc_info;
    /* Pointers to the IP v4/v6 addresses. */
    npf_addr_t *  npc_srcip;
    npf_addr_t *  npc_dstip;
    /* Size (v4 or v6) of IP addresses. */
    int           npc_ipsz;
    size_t        npc_hlen;
    int           npc_next_proto;
    /* IPv4, IPv6. */
    union {
        struct ip      v4;
        struct ip6_hdr v6;
    } npc_ip;
    /* TCP, UDP, ICMP. */
    union {
        struct tcphdr  tcp;
        struct udphdr  udp;
        struct icmp    icmp;
    } npc_l4;
} npf_cache_t;
```



Current TODO list

- ▶ Replace red-black trees with Patricia (radix) trees for tables



Current TODO list

- ▶ Replace red-black trees with Patricia (radix) trees for tables
- ▶ Pregenerate an array for masks



Current TODO list

- ▶ Replace red-black trees with Patricia (radix) trees for tables
- ▶ Pregenerate an array for masks
- ▶ Support per-rule statistics



Current TODO list

- ▶ Replace red-black trees with Patricia (radix) trees for tables
- ▶ Pregenerate an array for masks
- ▶ Support per-rule statistics
- ▶ Multiple address per interface support on interface-based rules
- ▶ Dynamic interface tracking
 - ▶ Let's say you have a rule based on an interface instead of an address...
 - ▶ ...you compile and load the rules...
 - ▶ ...then change the interface's address?
- ▶ Control fragmentation on a per-interface basis



Current TODO list

- ▶ Replace red-black trees with Patricia (radix) trees for tables
- ▶ Pregenerate an array for masks
- ▶ Support per-rule statistics
- ▶ Multiple address per interface support on interface-based rules
- ▶ Dynamic interface tracking
 - ▶ Let's say you have a rule based on an interface instead of an address...
 - ▶ ...you compile and load the rules...
 - ▶ ...then change the interface's address?
- ▶ Control fragmentation on a per-interface basis
- ▶ Stateful NAT64 [RFC 6146]
- ▶ IPv6-to-IPv6 Network Prefix Translation (NPTv6) [RFC 6296]



Current TODO list

- ▶ Replace red-black trees with Patricia (radix) trees for tables
- ▶ Pregenerate an array for masks
- ▶ Support per-rule statistics
- ▶ Multiple address per interface support on interface-based rules
- ▶ Dynamic interface tracking
 - ▶ Let's say you have a rule based on an interface instead of an address...
 - ▶ ...you compile and load the rules...
 - ▶ ...then change the interface's address?
- ▶ Control fragmentation on a per-interface basis
- ▶ Stateful NAT64 [RFC 6146]
- ▶ IPv6-to-IPv6 Network Prefix Translation (NPTv6) [RFC 6296]
- ▶ Write more documentation :)



What we haven't covered today and why...

- ▶ Performance testing
 - ▶ No point in in-place testing, since the locking of the network stack is not fine-grained enough
 - ▶ Will do userspace tests once the current codebase is merged
- ▶ Packet sniffing



Questions and answers?



Thank you!

