

Design and Implementation of an IPv6 Plugin for the Snort Intrusion Detection System

Martin Schütte



22. September 2011

IPv6 als Sicherheitsproblem

Intrusion Detection Systems and Snort

Snort APIs

Snort IPv6 Plugin

Tests

Fazit

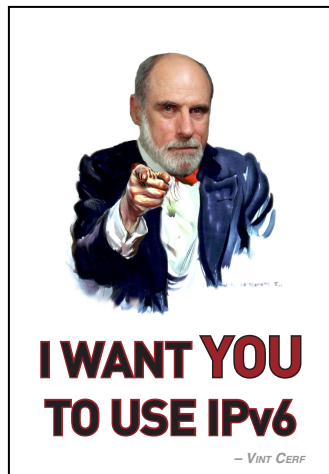
Stand ~ 1994

- IPv4-Internet: Forschungs- und Uni-Netze
- bekannte Design- & Implementierungs-Fehler
- wenig Erfahrung mit Protokoll-Sicherheit
- wenig Druck zur Verbesserung



Stand ~ 2011

- IPv6-Internet: Forschungs- und Uni-Netze
- bekannte Design- & Implementierungs-Fehler
- wenig Erfahrung mit Protokoll-Sicherheit
- wenig Druck zur Verbesserung



www.cs.brown.edu/~vint/cerf/

IPv6 Probleme

- RFCs von 1995/1998
- ⇒ 15 Jahre IPv4-Sicherheits-Erfahrung nachzuholen
- viele Internet-Drafts (IPsec, SEND, ...)
 - wenig Implementierungen
 - fast nichts in Endgeräten

Angriffe auf IPv6

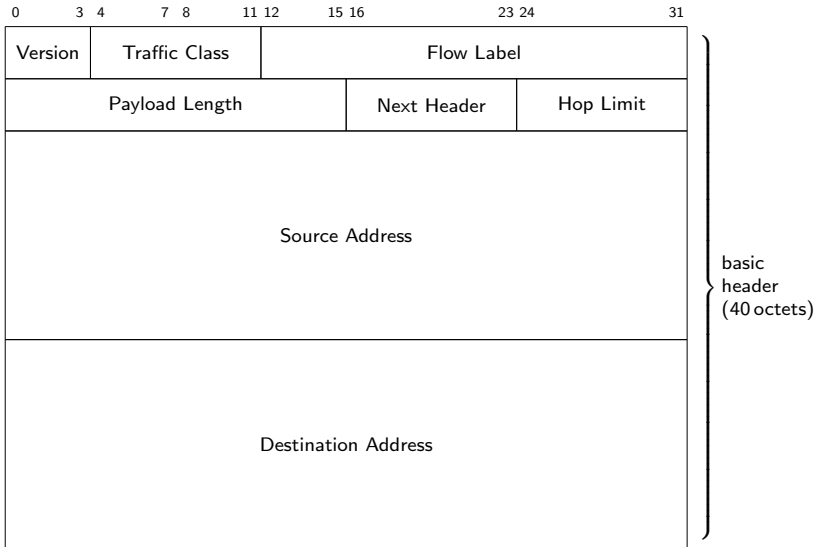
Das übliche:

- Wertebereiche für Felder
- Fragmentierung
- Denial of Service
- Portscans
- Fehler in Anwendungsschicht

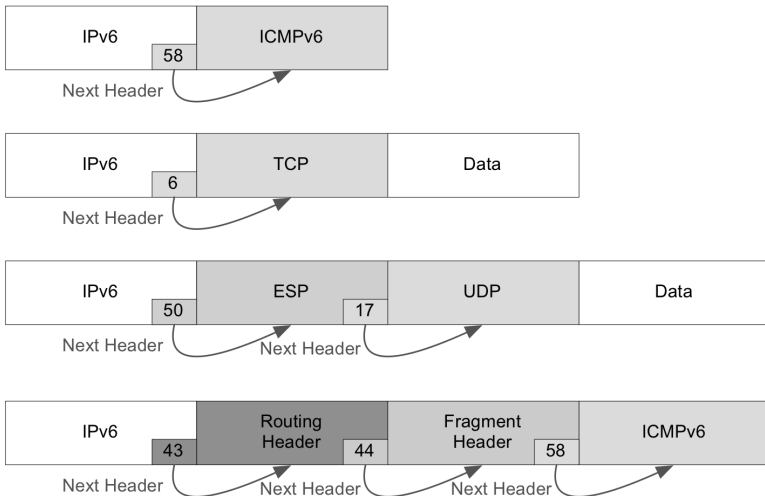
IPv6-spezifisch:

- variable Header
- Autokonfiguration
- Multicast
- Routing
- v4/v6-Transition

IPv6 Header Format



IPv6 Extension Header



Destination/Hop-by-Hop Option Header

0	7 8	15 16	23 24	31
Next Header	Hdr Ext Len	Opt Type	Opt Len	
Opt Value	...			

0	7 8	15 16	23 24	31
Next Header: 0x3a <i>ICMPv6</i>	Hdr Ext Len: 0x00 <i>8 octets</i>	Opt Type: 0x05 <i>Rtr alert</i>	Opt Data Len: 0x02 <i>2 octets</i>	
Opt Data: 0x00 0x00 <i>MLD</i>		Opt Type: 0x01 <i>PadN</i>	Opt Data Len: 0x00 <i>0 octets</i>	

Autokonfiguration und Neighbor Discovery

Design-Prämisse: sicheres und vertrauenswürdiges LAN

einfacher Denial of Service:

1. Host Alice startet *Duplicate Address Detection*
„Benutzt jemand die IP X?“
2. Host Eve antwortet „Ich benutze IP X.“
3. goto 1

Routing/Man in the Middle:

1. Host Eve sendet ICMPv6 Redirect
„Hier Router Bob, für *google.com* bitte Router Eve benutzen.“

Routing und Transition

Routing:

- Umfangreiche Spezifikation
- RH0 (source routing) deprecated
- RH2 für MobileIPv6 nötig

Transition:

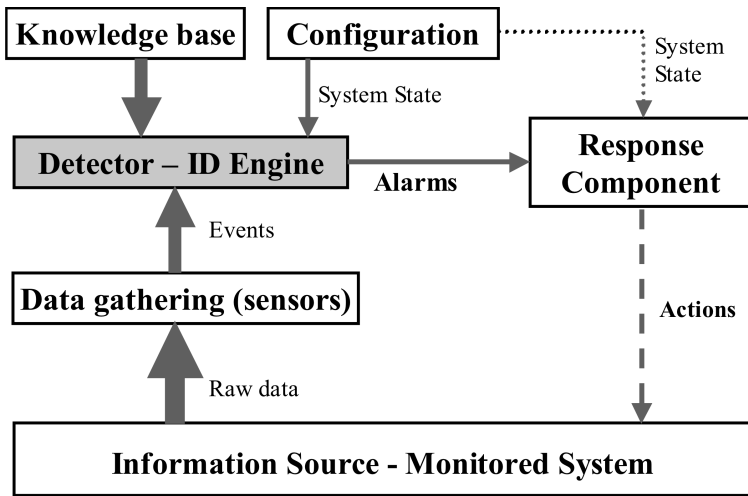
- Dual-Stack: zwei Paketfilter
- Tunnelling: leichte Filter-Umgehung
- Automatic Tunnel Routing Loops

Angriffs-Sammlung: THC Toolkit

Tools/Angriffe/Tests für:

- Autoconfiguration DoS
- Neighbour Cache
- Routing/Redirect
- Flood-Attacks
- Multicast Listener Discovery
- DHCPv6
- `implementation6`

(Network) Intrusion Detection System

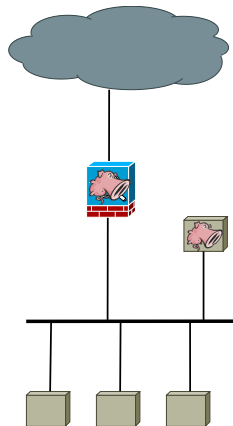


Lazarević et al, Intrusion Detection: A Survey, 2005

Zielsystem: Snort 2.9.1



- verbreitetes Open Source NIDS
- Packet-Sniffer (IDS)
- Filter-/inline-Modus
(*Intrusion Prevention System*)
- Plugin APIs
- Dekoder für gängige Tunnelprotokolle



IPv6 Support

im Prinzip Ja, aber ...

Alle relevanten IDS haben IPv6-Support.

Aber was heißt das?

- Fragment-Reassemblierung
- TCP & UDP Dekodierung
- Decoder-Warnung bei groben Protokollfehlern

(Noch?) nicht:

- neue Erweiterungen (Routing Header, Jumbograms)
- alle Regeloptionen (fragbits)
- IPv6-spezifischer Angriffe (ICMPv6/neighbor discovery)

IPv6 Signaturen

Viele Protokollfelder und Regeln abwärtskompatibel.

```

alert ip icmp any -> any any \
  (msg:"IPv6 ICMP Echo-Request?"; itype:128; \
  classtype:icmp-event; sid:2000001; rev:1;)

```

```

alert icmp any any -> any any \
  (msg:"ICMP ID 0xdead (THC?)"; icmp_id: 57005; \
  sid:124860; rev:1;)

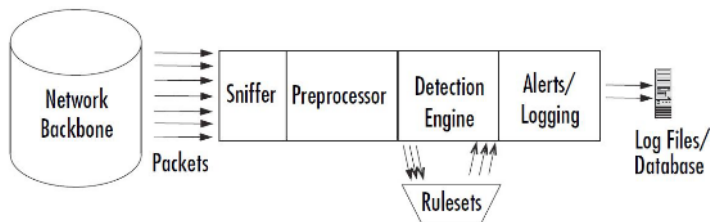
```

Keine Schlüsselwörter für neue IPv6-Felder.

Kein Weg nur-IPv6 Regeln zu schreiben.

⇒ eigenes IPv6-Plugin

Snort-Architektur



- Dynamic Rule API
- Dynamic Preprocessor API

Snort Dynamic Rule API

- genau eine Regel als dynamische Bibliothek (.so)
 - Funktionsaufruf mit Rückgabe `match/no_match`
- ⇒ nicht weiter betrachtet

Snort Dynamic Preprocessor API

- dynamische Bibliothek (.so)
- aus `snort.conf` geladen, aktiviert, konfiguriert
- Präprozessor und/oder Regeloptionen

Snort Dynamic Preprocessor Plugin

Funktionsweise:

- Registriert Callback für Pakete von Typ IP/ICMP/UDP/TCP/...
- Zugriff auf Snort-Teilsysteme
- Kann Zustand halten
- Kann Logs/Alarmer erzeugen
- Kann Regeloptionen implementieren

IPv6 Präprozessor

Funktionsweise:

- Liest ICMPv6-Nachrichten
- Verfolgt Netz-Zustand, d. h. (MAC, IP) von
 - Routern
 - Hosts
 - laufenden DADs
- Alert bei neuen Hosts, Router-Änderungen u. ä.

Konfiguration

in snort.conf

```
preprocessor ipv6: \
    net_prefix 2001:0db8:1::/64 \
    router_mac 00:16:76:07:bc:92 \
    host_mac ... \
    max_unconfirmed 32768 \
    max_routers 8 \
    expire_run 20 \
    keep_state 180 \
```

IPv6 Checks

SID	Message
1	RA from new router
2	RA from non-router MAC address
3	RA prefix changed
4	RA flags changed
5	RA for non-local net prefix
6	RA with lifetime 0
7	new DAD started
8	new host in network
9	new host with non-allowed MAC address
10	DAD with collision
11	DAD with spoofed collision
12	mismatch in MAC and NDP source linkaddress option
13	ipv6: extension header has only padding options (evasion?)
14	ipv6: option lengths != ext length

IPv6 Regeloptionen

Ziel:

- IPv6-Felder für Signaturen zugänglich machen
- Basis-Header, Erweiterungs-Header, Neighbor Discovery-Optionen

Funktionsweise:

- Callbacks für Options-Schlüsselwörter
- Aufruf mit Parametern und Paket
- Rückgabe `match/no_match`

IPv6 Regeloptionen

```

alert icmp any any -> any any (itype:8;  ipv: 4; \
  msg:"ICMPv4 PING in v4 pkt"; sid:100000; rev:1;)
alert icmp any any -> any any (itype:8;  ipv: 6; \
  msg:"ICMPv4 PING in v6 pkt"; sid:100001; rev:1;)

alert icmp any any -> any any (itype:128; ipv: 4; \
  msg:"ICMPv6 PING in v4 pkt"; sid:100002; rev:1;)
alert icmp any any -> any any (itype:128; ipv: 6; \
  msg:"ICMPv6 PING in v6 pkt"; sid:100003; rev:1;)

```

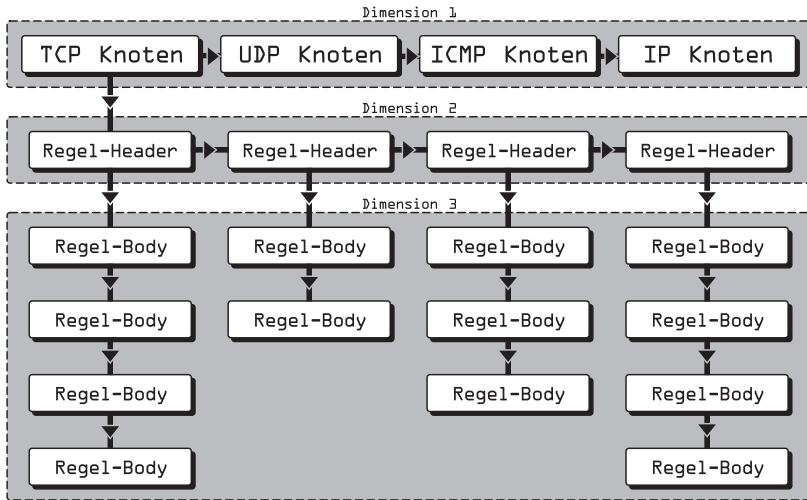
IPv6 Regeloptionen

```
alert ip any any -> any any (ip6_rh: !2; \
  msg:"invalid routing hdr"; sid:1000004; rev:1;)

event_filter gen_id 1, sig_id 1000004, type limit, \
  track by_dst, count 1, seconds 60

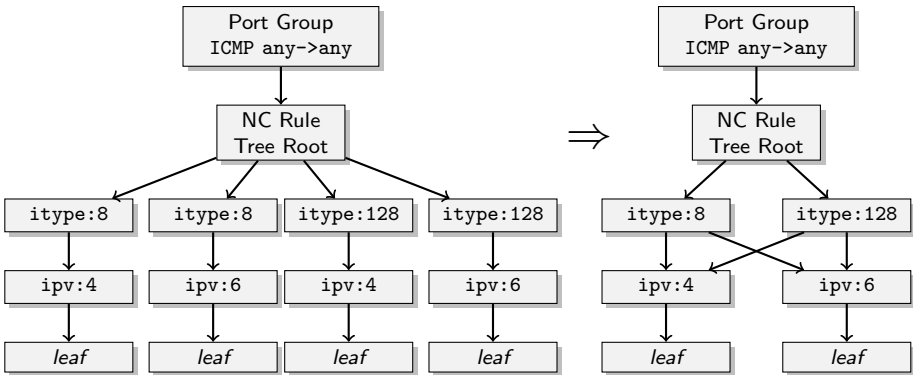
alert icmp any any -> any any (ipv: 6; itype: 134; \
  detection_filter: track by_dst, count 5, seconds 1; \
  msg:"ICMPv6/RA flooding"; sid:124850; rev:1;)
```

Snort Detection Engine (alt)



Bechtold/Heinlein, Snort, Acid & Co, 2004

Snort Detection Engine (optimiert)



Regeloptionen des IPv6-Plugins

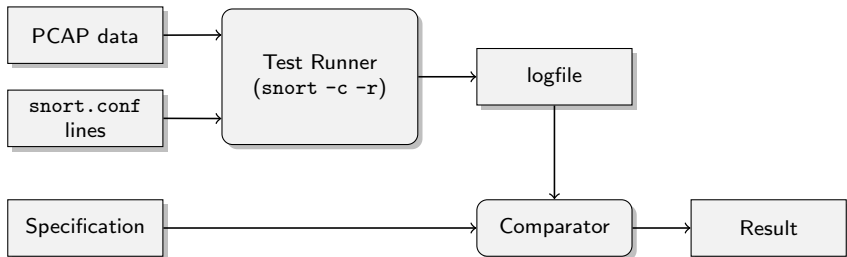
<code>ipv</code>	IP version
<code>ip6_tclass</code>	Traffic Class
<code>ip6_flow</code>	Flow Label
<code>ip6_exthdr</code>	Extension Header
<code>ip6_extnum</code>	Num. of Ext Hdrs.
<code>ip6_option</code>	Destination-/HbH-Option
<code>ip6_optval</code>	Destination-/HbH-Option Value
<code>ip6_rh</code>	Routing Header
<code>icmp6_nd</code>	Neighbor Discovery (bool)
<code>icmp6_nd_option</code>	Neighbor Discovery Option

Funktionaler Test

Snort-Funktionen gut zu testen:

- Eingabe:
 - PCAP-Datei (anstatt Netzwerk-Input)
 - `snort.conf`
- Ausgabe:
 - Log-Ereignisse und Alarme

tester.pl



Beispiel-Tests

```
-
test: sendpees6
pcap: sendpees6_1sec.pcap
conf: simple.conf
spec: "[1:124806:1] , [1:124851:1] , [248:12:1] "
```

```
-
# ping with empty hbh ext (i.e. only padding)
test: ping_padding
pcap: ping_hbh_pad.pcap
conf: simple.conf
spec: "[116:432:1] , [248:13:1] "
```

Performance

- Zustandslose Checks sind schnell:
Plugin liest `struct SFSnortPacket`
- Zustand verfolgen kostet Zeit und Speicher:
⇒ DoS-Gefahr, daher Limits

⇒ ähnlich wie andere Plugins (SSL, SMTP, ...)

⇒ Snort-Dekoder ist Bottleneck

Snort-Funktionen

IPv6-fertige Snort-Komponenten

- Portscans (sfportscan) & Fragmentierung (frag3)
- Paketfilter (Inline-Modus, je nach DAQ)
- Logging (unified2)
- nach und nach mehr Dekoder-Checks

Praktische Probleme

- kein Snort-SQL-Schema für IPv6-Events
⇒ keine IPv6-Events in barnyard2-DB-Output, BASE, Snorby, u. ä.
- Präprozessor-/Dekoder-Alarme nicht eindeutig

Praktische Probleme

Idee:

```
alert ip any any -> any any (ipv: 6; \
  ip6_exthdr: 44; fragbits: !M; fragoffset: 0; \
  msg:"IPv6: redundant fragment hdr (THC?)" ; \
  sid:124860; rev:1;)
```

aber mit Decoder Alerts:

123:8:1 frag3: Fragmentation overlap

123:10:1 frag3: Bogus fragmentation packet. Possible BSD attack

Grenze der Erkennung

Grundlegende Einschränkung: Layer 2/Ethernet

- ⇒ keine sichere Host-ID
- ⇒ kein Filter

Unsicherer Layer 2 nicht auf Layer 3 zu sichern.

Fazit

- Plugin funktioniert
 - Als dynamische Bibliothek installierbar
 - Grundlage für neue Signaturen
 - Snort & Plugin erkennen THC-Angriffe
- ⇒ jetzt Praxistest in realen Netzen