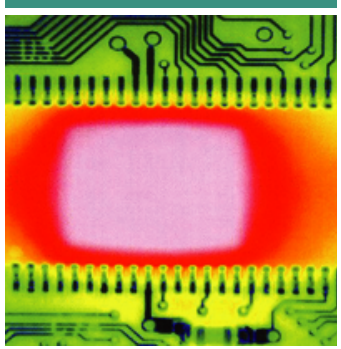
ft6-Testergebnisse der Juniper J2320

Sven Schindler

Beuth Hochschule Berlin





Gliederung

Test-Setup

- Cient mit Ubuntu 12.04 Kernel 3.2.0-40-generic-pae
- Server mit Ubuntu 12.04 Kernel 3.2.0-31-generic-pae
- Python 2.7.3 und Scapy 2.2.0
- Juniper J2320 mit Junos JSR 12.1R5.5-export

Übersicht

Test	Default config	Recommended config
ICMPv6 Filtering	✗	✓
Routing Header	✗	✗
Header Chain	✗	✗
Overlapping Fragments	✗	✗
Tiny Fragments	✗	✗
Excessive Hop-By-Hop Options	✗	✗
PadN Covert Channel	✗	✗
Address Scopes	✓	✓

Ergebnisse - Test 1: ICMPv6 Filtering

Prüfe, ob Firewall ICMPv6-Pakete korrekt weiterleitet oder verwirft. Die Pakete sind in Kategorien Mandatory Filter, Optional, Nonfiltered eingeteilt.

default config

- ✓ ICMPv6-Types 130, 131, 132 wurden verworfen (Multicast Listener Query/Report/Done)
- ✗ alle anderen Pakete wurden an das Ziel weitergeleitet

improved config

- ✓ Mandatory Filter: alle verworfen
- ✓ Optional: alle verworfen
- ✓ Nonfiltered: alle weitergeleitet

Ergebnisse - Test 2: Routing Header

Prüfe, ob Firewall Pakete mit Routing Header korrekt weiterleitet oder verwirft. Beachte dabei den Routing Header Type und Wert des segments-left Feldes.

default config

- ✓ Type = 0, segments-left $=$ 0: weitergeleitet
- ✗ Type = 0, segments-left $\neq$ 0: weitergeleitet
- ✗ Type = 2, segments-left $\neq$ 1: weitergeleitet
- !! Type = 2, segments-left $=$ 1: weitergeleitet
- !! Type = 200, segments-left $=$ 0: weitergeleitet
- ✗ Type = 200, segments-left $\neq$ 0: weitergeleitet

$\rightarrow$ Keine Firewallregel für Routing Header definiert

Ergebnisse - Test 2: Routing Header

Neukonfiguration

```
term routing_header_reject {
        from {
                next-header routing;
        }
        then {
                reject;
        }
}
```

→ Keine Filterregel für bestimmte Routing Header Typen verfügbar

Ergebnisse - Test 2: Routing Header

Empfohlene Konfiguration hängt von Mobility Support ab.

improved config - wenn kein Mobility Support benötigt wird

- ✗ Type = 0, segments-left $=$ 0: verworfen
- ✓ Type = 0, segments-left $\neq$ 0: verworfen
- ✓ Type = 2, segments-left $\neq$ 1: verworfen
- ✗ Type = 2, segments-left $=$ 1: verworfen
- ✗ Type = 200, segments-left $=$ 0: verworfen
- ✓ Type = 200, segments-left $\neq$ 0: verworfen

$\rightarrow$ Keine Firewallregel für Routing Header definiert

Ergebnisse - Test 3: Extension Header Chain

Prüfe, ob Firewall Pakete mit diversen Header Chains korrekt weiterleitet oder verwirft. Valide Kombinationen sind in RFC 2460 definiert.

default config

- ✓ DSTOPT: weitergeleitet
- ✓ HBH: weitergeleitet
- ✗ DSTOPT-HBH: weitergeleitet
- ✗ DSTOPT-DSTOPT: weitergeleitet
- ✓ HBH-HBH: verworfen
- ✓ DSTOPT-RH-DSTOTP: weitergeleitet
- ✓ HBH-DSTOPT-RH-HBH: verworfen

→ Keine Möglichkeit nach bestimmten Header Chains zu filtern

Ergebnisse - Test 4: Overlapping Fragments

Prüfe, ob Firewall fragmentierte Pakete korrekt weiterleitet oder verwirft. Beachte Fragmente mit inkorrektem Offset, d.h. Fragmente, die “frühere” Informationen überschreiben.

default config

- ✓ korrektes Offset: weitergeleitet
- ✗ inkorrektes Offset (Port überschreiben): weitergeleitet
- ✗ inkorrektes Offset (Payload überschrieben): weitergeleitet

→ Keine Möglickeit dieses Verhalten durch Filter zu verbessern

Ergebnisse - Test 5: Tiny Fragments

Prüfe, ob Firewall Pakete mit “Tiny Fragments” korrekt weiterleitet oder verwirft. Tiny Fragments haben keinen TCP/UDP-Header im ersten Fragment, eine Entscheidung kann also erst nach Reassembly getroffen werden. Prüfe außerdem, ob die Firewall lange genug auf “verspätete” Fragmente wartet.

default config

- ✓ Port 80 im zweiten Fragment: weitergeleitet
- ✗ Port 22 im zweiten Fragment: weitergeleitet
- ✓ Nach 55 Sekunden eintreffendes Fragment: weitergeleitet
- ✗ Nach 65 Sekunden eintreffendes Fragment: weitergeleitet

→ Es werden alle Pakete weitergeleitet, Drop aller fragmentierten Pakete möglich