HoneydV6
A low-interaction IPv6 honeypot

Sven Schindler

Potsdam University / Beuth Hochschule Berlin

Berlin, June 12, 2013

Outline

What is a virtual honeypot and why do we need it?

Honeypot definition

A virtual honeypot is a security device that has no production value [2].This can be something like a computer or even a mobile phone which only purpose is to attract attackers, so that their attacks can be analysed.

- provides level of interaction
- classification based on level of interaction

Honeyd

- open source low-interaction honeypot
- latest release version 1.5c does not support IPv6
- custom network stack
- simulate entire networks
- supports OS fingerprinting
- provides framework for service scripts
- Tiny Honeypot, SCADA HoneyNet Project based on Honeyd

Honeyd architecture

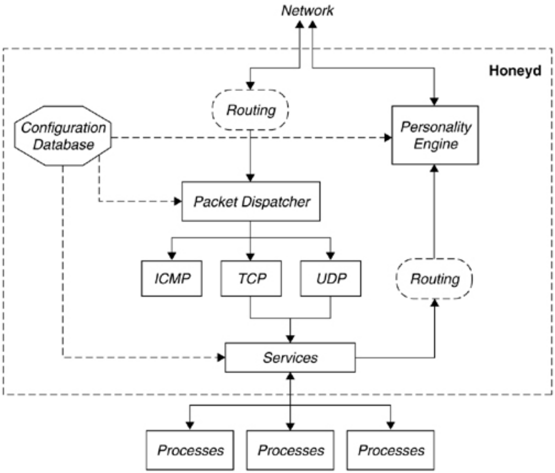

Figure: Honeyd architecture [1]

Adapting the configuration of virtual hosts

- configuration parser modified to accept IPv6 addresses
- IPv6 and IPv4 templates managed in splay tree

Example IPv4 configuration

```
create windows
set windows default tcp action reset
add windows tcp port 21 "scripts/ftp.sh"

set windows ethernet "aa:00:04:78:98:76"

bind 192.168.1.5 windows
bind 192.168.1.6 windows
```

Modifying packet processing

- template searched for incoming packets
- packets assembled and forwarded to IPv6 dispatcher
- fragmentation length and offset logged
- checksum calculations updated
- get_ip6_next_hdr implemented
- updated callbacks to tcp_recv_cb46 and udp_recv_cb46
- TCP/UDP connection structures updated

Implementing ICMPv6 and the Neighbor Discovery Protocol

- ICMPv6 echo request/reply
- ICMPv6 Time Exceeded and Destination Unreachable
- send and process neighbor solicitations
- send router solicitations
- process router advertisements

Pitfalls

scope IDs in link-local addresses

```
static void addr_remove_scope_id(struct addr* ip6)
    {
  if (ip6->addr_data8[0]==0xfe && ip6->addr_data8
      [1]==0x80) {
    /* delete scope id */
    ip6->addr_data8[2]=0;
    ip6->addr_data8[3]=0;
  }
}
```

Pitfalls

use of dynamic arrays

```
struct interface {
  TAILQ_ENTRY(interface) next;

  struct intf_entry if_ent;
  int if_addrbits;
  struct event if_recvev;
  pcap_t *if_pcap;
  eth_t *if_eth;
  int if_dloff;

  char if_filter[1024];
};
```

Random IPv6 request processing

- linear IPv6 address scan is impossible
- attacker needs to find hosts
- dynamically create new virtual hosts on demand
- observe new scan approaches
- all login attempts logged

Configuration of random IPv6 request processing

Configuration

```
create randomdefault
set randomdefault default tcp action reset
add randomdefault tcp port 21 "scripts/ftp.sh"
add randomdefault tcp port 80 "scripts/web.sh"
set randomdefault ethernet "aa:00:04:78:98:78"

randomipv6 0.5 randomdefault 256

randomexclude 2001:db8::1
randomexclude 2001:db8::2
randomexclude 2001:db8::3
```

Performance tests - throughput measurements

- PRIMERGY TX200 S5 Server with an Intel Xeon processor 5500 series and 4096 MB of RAM running Ubuntu 12.04
- benchmark client was installed on a Lenovo ThinkPad L520 with an Intel i5-2450M CPU and 4096 MB of RAM
- computers connected via Brocade FWS648G FastIron switch using Gigabit Ethernet

Filesize	1.5c (IPv4)	V6 (IPv4)	V6 (IPv6)
50 MB	15.98 s	16.19 s	16.33 s
100 MB	31.85 s	31.94 s	32.36 s

Table: Comparison of transmission time in seconds between the original Honeyd version 1.5c and HoneydV6 - median values of 5 test runs

Performance tests - HTTP get request measurements

- generated log file containing 20,000 HTTP GET request from different source addresses
- 600 requests per second
- honeyd configured to simulated single hosts (IPv4 and IPv6 connected)
- web.sh script on port 80

1.5c (IPv4)	V6 (IPv4)	V6 (IPv6)
212.57	214.00	205.75

Table: Comparison of the number of HTTP GET requests per second that Honeyd 1.5c and HoneydV6 is able to handle without any packet loss.

Conclusion and future work

- HoneydV6 is first low-interaction honeypot which can simulate entire IPv6 networks on a single host
- may be used to add IPv6 support for low-interaction honeypots based on honeyd
- IPv4 / IPv6 search replace was not sufficient
- new protocols implemented (NDP, ICMPv6)
- random IPv6 request processing helps to understand new scan approaches
- OS fingerprinting and tunnel support not yet implemented
- working on shellcode detection engine

References

Niels Provos and Thorsten Holz.
Virtual Honeypots - From Botnet Tracking to Intrusion Detection.
Addison-Wesley, 2008.

Christian Seifert, Ian Welch, and Peter Komisarczuk.
Taxonomy of honeypots.
Technical report, Victoria University of Wellington, Wellington, 2006.