



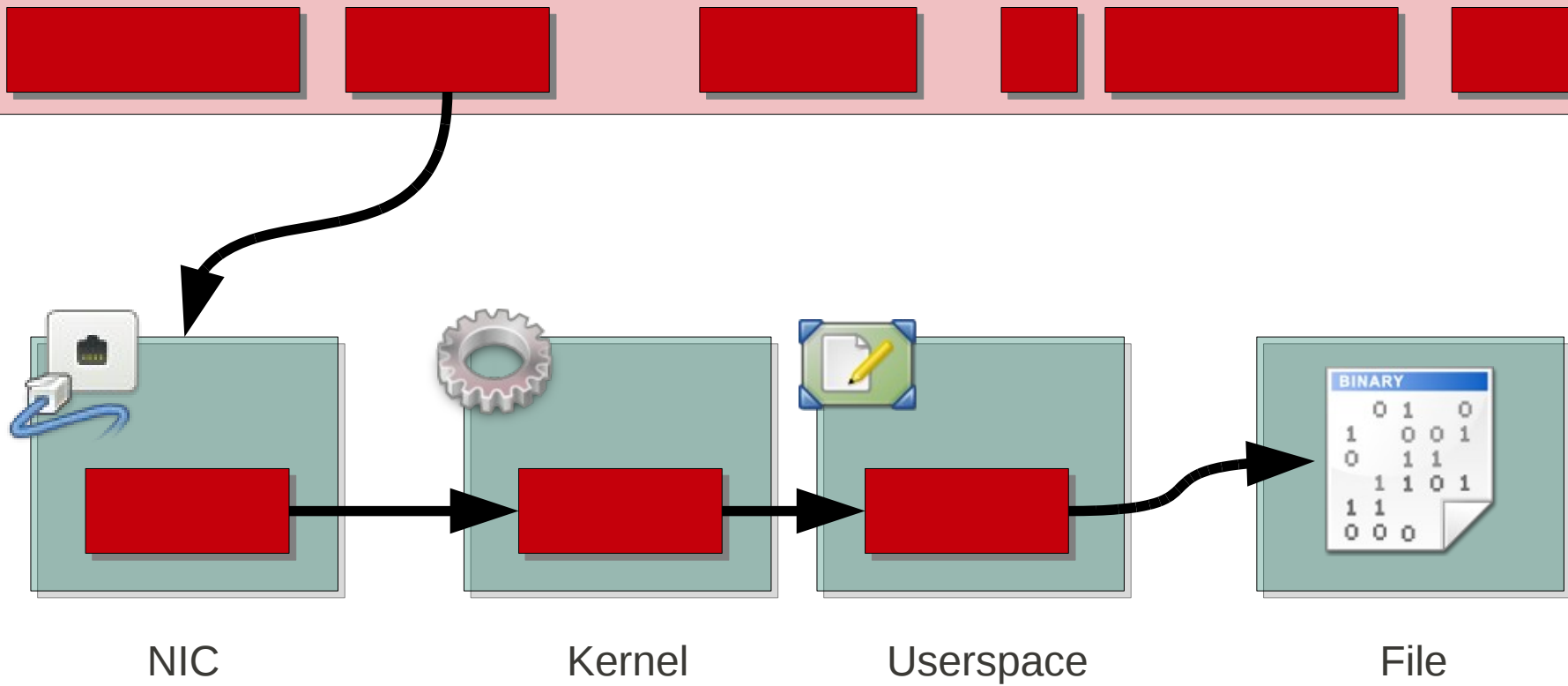
High-Performance Packet Sniffing and Traffic Mining

Tillmann Werner, Senior Virus Analyst, Kaspersky Lab
HoneyNet Workshop 2011, Public Day
Paris, 21 March, 2011

High Performance Packet Sniffing

multicap

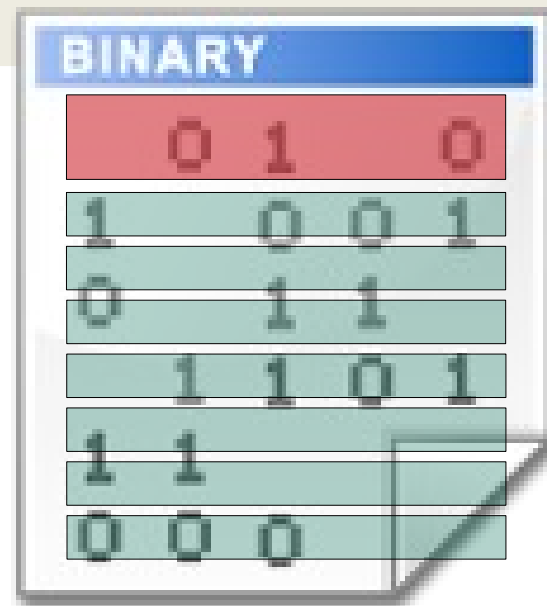
Packet Sniffing



The Pcap File Format

Straight-Forward File Format

- ▶ Portable library for packet sniffing
- ▶ Convenient API for programmers
 - Live capturing
 - Writing and reading dump files
- ▶ Open source, GPLv2
- ▶ Used by tools like tcpdump, Wireshark, Snort, ...
- ▶ Time resolution in microseconds



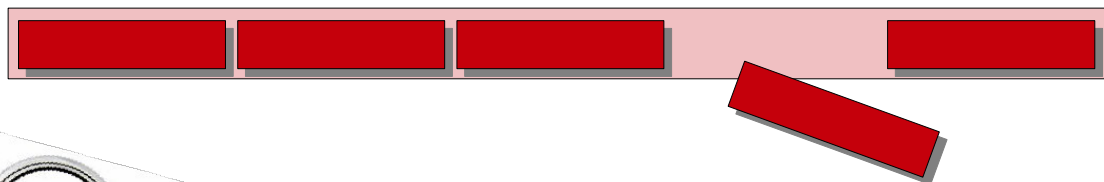
```
struct pcap_file_header {  
    bpf_u_int32 magic;  
    u_short version_major;  
    u_short version_minor;  
    bpf_int32 thiszone;  
    bpf_u_int32 sigfigs;  
    bpf_u_int32 snaplen;  
    bpf_u_int32 linktype;  
};
```

```
struct pcap_pkthdr {  
    struct timeval ts;  
    bpf_u_int32 caplen;  
    bpf_u_int32 len;  
};
```

Do Not Drop The Packets

Packet Drops

- ▶ Sniffer too slow: packet drops
- ▶ Lost information cannot be recovered
- ▶ Missing packets can render TCP streams unusable



Sniffing Performance

- ▶ Allocating, copying and freeing memory takes time
- ▶ Getting the system time costs CPU cycles
- ▶ Reduce such calls as much as possible

Designing *multicap*

Minimize Memory Allocations

- ▶ Use a `PF_PACKET` socket
- ▶ Attach a user-space ring buffer with `setsockopt(PACKET_RX_RING)`
- ▶ This is Linux only

No System Calls To Get Packet Times

- ▶ `PF_PACKET` already stores the time stamp in the packet struct
 - nano-second time resolution without further system calls
- ▶ No need to call `localtime()` etc.

Memory-mapped Dump Files

- ▶ `mmap()` for increased dumping
- ▶ Pre-allocate multiples of page size



Configuration Example

```
rotate = "1d" // "h|d|G|M"
path = ""
file = "$tracker-%Y%m%d.pcap"

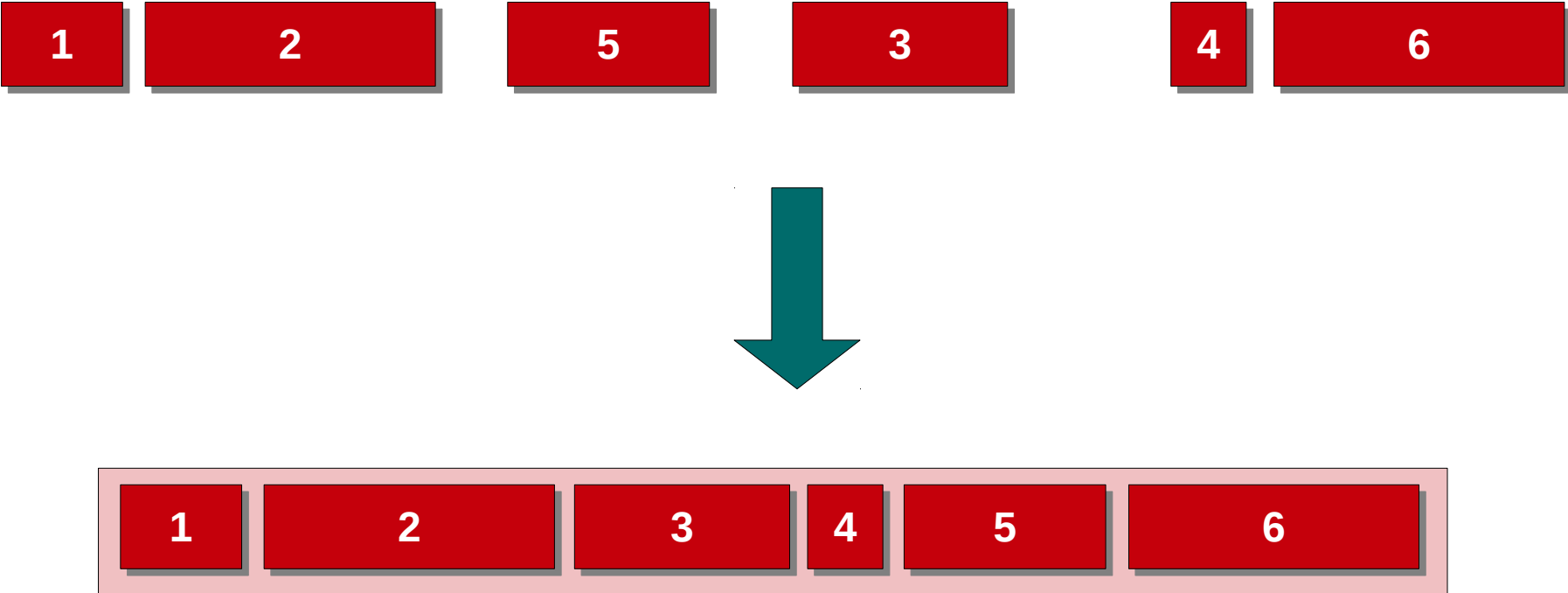
trackers = {
    tcponly =
    {
        enabled = "true"
        interface = "eth0"
        bpf = "tcp"
        snaplen = "0"
        rotate = "10M"
        path = "var/log/multicap/$tracker/%Y-%m-%d"
        file = "%H:%M:%S.pcap"
        promisc = "0"
    }
}
```

multicap Demo

Packet Trace File Processing

streams

Stream Reassembly



TCP Stream Reassembly

TCP Streams

- ▶ Stream: reliable, ordered stream of data
- ▶ OS assembles segments in the right order



Stream Reassembly Tools

- ▶ Wireshark
- ▶ tcpick
- ▶ tcpflow

The screenshot shows the Wireshark interface with a packet capture of TCP traffic. The packet list pane shows 8 packets, all from source 127.0.0.1 to destination 127.0.0.1. The selected packet (No. 8) is a TCP segment. The packet details pane shows the following layers: Ethernet II, Internet Protocol, and Transmission Control Protocol. A context menu is open over the selected packet, with the following options: Mark Packet (toggle), Set Time Reference (toggle), Apply as Filter, Prepare a Filter, Conversation Filter, Colorize Conversation, SCTP, Follow TCP Stream (highlighted), Follow UDP Stream, Follow SSL Stream, Copy, Decode As..., Print..., and Show Packet in New Window.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	48618 → search_agent [SYN] Seq=0 Win=32768
2	0.000018	127.0.0.1	127.0.0.1	TCP	48618 → search_agent [ACK] Seq=0 Ack=1 Win=0
3	0.000038	127.0.0.1	127.0.0.1	TCP	48618 → search_agent [ACK] Seq=1 Ack=1 Win=0
4	0.000084	127.0.0.1	127.0.0.1	TCP	48618 → search_agent [ACK] Seq=1 Ack=52 Win=0
5	0.000093	127.0.0.1	127.0.0.1	TCP	48618 → search_agent [ACK] Seq=1 Ack=52 Win=0
6	0.000123	127.0.0.1	127.0.0.1	TCP	48618 → search_agent [ACK] Seq=1 Ack=52 Win=0
7	0.000210	127.0.0.1	127.0.0.1	TCP	48618 → search_agent [ACK] Seq=1 Ack=52 Win=0
8	0.000236	127.0.0.1	127.0.0.1	TCP	48618 → search_agent [ACK] Seq=1 Ack=52 Win=0

TCP Stream Reassembly



TCP Streams

- ▶ Stream: reliable, ordered stream of data
- ▶ OS assembles segments in the right order

Stream Reassembly Tools

- ▶ Wireshark
- ▶ tcpick
- ▶ tcpflow

No.	Time	Source	Destination	Protocol	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	48618 > search_agent [SYN] Seq=0 Win=32768
2	0.000018	127.0.0.1	127.0.0.1	TCP	48618 > search_agent [ACK] Seq=0 Ack=1
3	0.000038	127.0.0.1	127.0.0.1	TCP	48618 > search_agent [ACK] Seq=0 Ack=1

Stream Content

```
00000000 1f 8b 08 00 b2 96 86 4d 00 03 0b c9 c8 2c 56 00 .....M .....V.
00000010 a2 44 85 e4 fc dc 82 a2 d4 e2 e2 d4 14 85 10 e7 .D.....
00000020 00 85 e2 92 a2 d4 c4 5c 3d 2e 00 1c ba 00 28 21 .....\ =.....(!
00000030 00 00 00 ...
```

Find Save As Print Entire conversation (51 bytes) [v] ASCII EBCDIC Hex Dump C Arrays Raw

Filter Out This Stream Close

Designing streams

Stream

- ▶ IP addresses, port numbers, initial sequence number

Stream Reassembly Strategy

- ▶ A SYN segment starts a new stream
- ▶ A RST or FIN segment terminates a stream
- ▶ Any segment gets copied at the right offset according to its sequence number

Interactive Command Line Tool

- ▶ Listing, counting, filtering, selecting, ... streams
- ▶ Easy integration of external tools

Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection

<http://insecure.org/stf/secnet_ids/secnet_ids.html>

streams Demo

Code

Where To Get Them

```
lftp anonymous@ftp.carnivore.it: /> cd projects/streams/  
lftp anonymous@ftp.carnivore.it: /projects/streams> ls  
-rw-r--r--  1 33      33      125549 Mar 16 16:19 streams-0.1.0.tar.bz2  
-rw-r--r--  1 33      33      235740 Mar 16 16:19 streams-0.1.0.tar.gz  
lftp anonymous@ftp.carnivore.it: /projects/streams>
```



carnivore.it code browser

source web interface to carnivore.it projects

index

Name

projects

carniwwhore
dionaea
honeytrap
libemu
liblcfg
libxmatch
multicap
nebula
nepenthes
pehunter
regoogle
streams

Description

webif for dionaeas xmpp fed postgres database
low interaction honeypot - our nepenthes sucesor
low interaction honeypot - the attacker implements the protocol
x86 emulation library to detect and emulate shellcode
lightweight configuration parsing library
library for matching patterns in xor-encoded data.
multi interface networkstream dump daemon
attack signitures via clustering
low interaction honeypot - use dionaea instead
snort plugin to carve PE files from network streams
google assisted reverse engineering
Play with pcap files

users

users/common/aguri
users/common/prosody
users/gento/dionaea
users/phibo/dionaea

aguri fork for the ladies
prosody fork with sensor network patches
dionaea branch for new things
dionaea branch for reason

<http://src.carnivore.it>

<ftp.carnivore.it>

Thank You

High-Performance Packet Sniffing and Traffic Mining

Tillmann Werner, Senior Virus Analyst, Kaspersky Lab
HoneyNet Workshop 2011, Public Day
Paris, 21 March, 2011