Attacking Highspeed Ethernet Links

Issued by:

ERNW GmbH
Enno Rey
Breslauer Strasse 28
D-69124 Heidelberg
www.ernw.de

Content

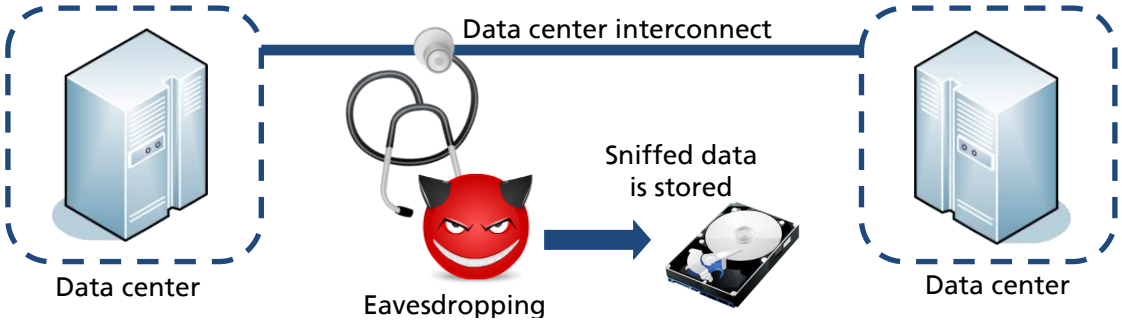
1 Introduction

There is a common misconception that the sheer amount of data coupled with multiplexed channels (e.g. WDM technology) makes successful eavesdropping attacks on high speed Ethernet – like 10 GbE or faster – unlikely. This is mainly based on the assumption that the amount of resources (e.g. RAM, [sufficiently fast] storage or CPU power) needed to process large files of captured data is a limiting factor. No practical evaluation of these assumptions has been performed so far though.

Therefore this paper aims to answer the following questions:

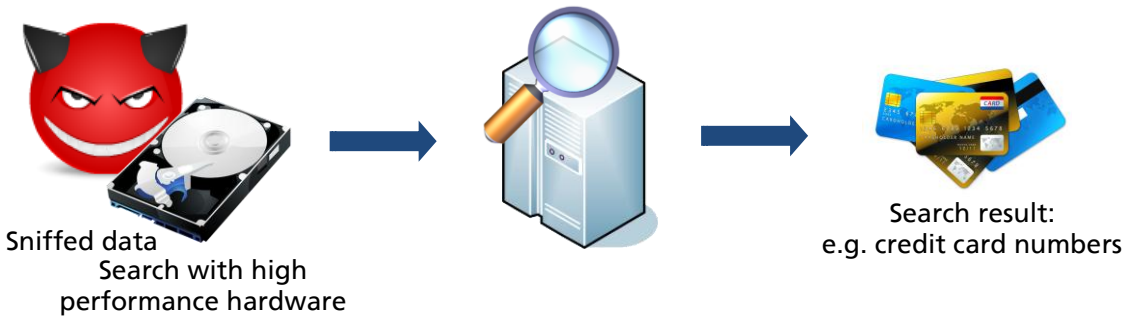
- Can the processing of large[1] amounts of captured data be done "in a feasible way"[2]?
- How much time and which type of hardware is needed to perform this task?
- Can this be done with readily available tools or is custom code helpful or even required? If so, how should that code operate?
- Can this task be facilitated by means of public cloud services?

To assess if the extraction of specific data from very large sets of captured network traffic can be done in a feasible way in terms of tools, hardware or time needed, a practical approach was undertaken. This included evaluating tools, building a lab with hardware fit for the purpose and performing a number of tests, including a demonstration of extracting credit card data from a file sized 500 GB and containing a virtual machine running an *Oracle* database. Such a file could, for example, derive from an eavesdropping attack against a high speed link connecting two data centers of an organization.



1.1 Prerequisites

It is assumed that the attacker has already gained access enabling her to eavesdrop on the high speed data link. A detailed description how this can be done can be found in [INFOGUARD_WP1][3]. It is further assumed the attacker has already performed the packet collection in standard pcap[4] format. The focus of the present paper is on the subsequent extraction of useful data from the dump file.



[1] *For demonstration purposes we limited our research to files in the 500 gigabyte size range (equivalent to the full live migration of 16 virtual machines with 32 GB of virtual memory each) which equates to ~ 400 seconds of sniffing on a fully saturated 10 GbE link.*
[2] *Where "feasible" means within some hours and by means of hardware to be purchased for less than 5,000 CHF.*
[3] *http://www.infoguard.com/docs/PDF/IG_Dokumente/WP_Fiber_Optic_Communication-e.pdf*
[4] *Pcap API - Libcap (Unix systems), www.tcpdump.org.*

2 Tackling the Problem

To execute the data extraction task in a feasible way the potential attacker has to dispose of the right tool(s) and hardware. This section lays out what is needed.

2.1 Tool pcap_extractor

As available command-line tools (like tethereal, tshark or tcpdump) have several limitations a special tool was developed. It's called *pcap_extractor*[5] and is basically the fastest possible implementation of a pcap file reader. It opens a libpcap file handle for the designated input file, applies a libpcap filter to it and loops through all the filter matching packets, writing them to an output pcap file. Contrary to tcpdump and most other libpcap based analysis tools, it provides the possibility to search for a given string inside of the matching packets, for example a credit card number or a username in easy way.

A sample call to search a pcap file for iSCSI packets which contain a certain credit card number and write them to the output file is shown in the following:

```
# pcap_extractor -i input-file.pcap -o output-file.pcap -f "tcp port 3260" -s "5486123456789012"
```

2.2 Hardware Needed

2.2.1 Identifying the bottleneck

While measuring the performance of multiple pcap analysis tools the profiling of system calls indicated that the tools spend between 85% and 98%[6] of the search time on waiting for I/O. This means that the fastest tool for 98% of the time doesn't do anything but waiting for dump data.

2.2.2 Actual lab setup with locally installed hardware

The lab system was designed to provide as much I/O bandwidth as possible and was composed of:

- Intel Core i7-990X Extreme Edition, 6x 3.46GHz
- 12GB (3 * 4GB) DDR3 1600MHz, PC3-12800
- ASRock X58 Extreme6 S1366 mainboard
- 4x Intel 510 Series Elm Crest SSD 250GB

The mainboard and the SSDs were chosen to support SATA3 with a theoretical maximal I/O bandwidth of 6 Gbps. FreeBSD was used as operating system.

2.2.3 Lab setup using the Amazon Elastic Compute Cloud (EC2)

The used Amazon EC2 instance was a so called *extra large instance.* Amazon describes this instance type as follows:

- 15 GB memory
- 8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each)
- 1,690 GB instance storage
- 64-bit platform
- I/O Performance: High
- API name: m1.xlarge

1 EC2 Compute Unit "provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor"[7], whereby the I/O performance scale of low, medium and high is not defined in more detail.

[5] *The tool can be downloaded from http://ernw.de/download/pcap_extractor.c.gz.*
[6] *Depending on the tool analyzed.*
[7] *As stated under http://aws.amazon.com/ec2/instance-types/.*

3 Results

In the following the test methodology and results are presented.

3.1 Description of overall test methodology

For the tests, five capture files were created using the *mergecap* utility. Different sample traffic dumps[8] were merged to five large files with different file sizes. All these files consist of several capture files containing a variety of protocols (including iSCSI and FCoE packets). More specifically capture files of ~40, ~80, ~200, ~500, and ~800 Gigabytes were created and were analyzed with different tools. At all tests the filtering expressions for tcpdump and pcap_extractor were configured to search for a specific source IP address and a specific destination IP address matching for some iSCSI packets inside of the capture file. The UNIX program *time* was used to measure the time of execution. Additionally the tools analyzing the data were started with the highest possible scheduling priority[9] to ensure execution with the maximum of available resources.

3.2 Test Results

The following table shows the most results achieved with the *pcap_exctractor* tool and having the collected data evenly spread across the four SSDs used in the local lab (see above).

Table 1: Results from test lab configured with four individual SSD drives and using custom tool

File size (GB)	Time of custom code (s)	Estimated time for 500 GB (s)	Time of tcpdump (s)	Estimated time for 500 GB (s)
44	113	*1284*	114	*1295*
80	203	*1269*	204	*1275*
204	521	*1277*	522	*1279*
404	1017	*1259*	1016	*1257*
516	1290	*1250*	1290	*1250*
808	2041	*1263*	2043	*1264*

This means that, in the demonstration setup, the search for specific credit card data could be performed in about 21 minutes, employing readily available tools and using common-of-the-shelf hardware for about 3500 CHF.

Furthermore the time needed scales in a linearly fashion with the file size, so that processing a 1 TB data volume presumably would have taken ~ 42 minutes, a 2 TB file would have taken ~ 84 minutes and so on. In addition, SSD prices are constantly declining, too.

3.2.1 Results from Tests in Amazon EC2 Cloud Setup

During the benchmarks, the performance was significantly lower than the one of the system described above even though eight different *Elastic Block Storage* (EBS) volumes were used to avoid the bottleneck of a single storage volume. Searching a 500 GB file with pcap_extractor took 4242 seconds (~ 71 minutes).

The overall performance of the test was apparently limited by I/O performance limitations within virtualized instances and virtualized storage systems.

[8] *http://wiki.wireshark.org/SampleCaptures*
[9] *This was done using the Unix nice command.*

4 Conclusions

It could be shown that the extraction of specific data from very large network traffic dumps can be achieved within a rather short time period. This was even possible using COTS hardware available for about 3500 CHF (as of March 2011). This means that an attacker disposing of (large) data sets resulting from previous eavesdropping attacks will most likely succeed in getting the exact data she's going after. In the lab setup, it took about 21 minutes to find a certain credit card number within a file sized 500 gigabyte which equates to a live migration of 16 virtual machines with 32 GB of virtual memory each.

The widespread perception that the sheer amount of data transferred over high speed network links prevents the extraction of data could thereby be proven wrong.

The entropy of a given file has practically no influence on the search/processing time needed. Furthermore it should be noted that the tests show a high correlation/degree of linearity between the size of the files (potentially derived by splitting one very large file into several chunks, each of them still of huge size) and the search time. So it seems there's a simple trade-off between the amount of storage provided and the search time. Given the ever-declining price of storage this will lead to even lower costs for such an attack in the future.

For all these reasons the protection of information, even when using optical networks, is vital and can be undertaken without restriction. InfoGuard is the leading manufacturer of layer 2 encryption solutions. These offer all-round and secure information exchange in MAN, WAN and SAN networks with 100% encryption throughput and minimum latency. Thus they are suitable for use in heavily utilized links and for time-critical applications. Data encryption is undertaken by the Advanced Encryption Standard (AES) with a key length of 256 bits.

5 Appendix

Demonstration of the Extraction of credit card data from an Oracle database running on virtual machine being live migrated

To show the impact of capturing and filtering data in business environments, an Oracle database was set up on a virtual machine. Figure 2 shows some sample content of the database, set up for simulating an environment storing sensible credit card data.

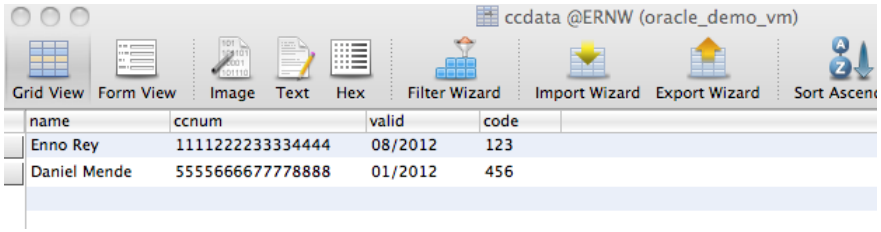


Figure 1: Contents of sample Oracle database

At first a transfer of the whole virtual machine running the Oracle database was initiated. For simplicity reasons, FTP was used instead of VMotion. By default both approaches do not encrypt the transferred data.
This capture file was then analyzed with pcap_extractor to search for a specific string (for example the name of a credit card holder). This is shown in the following figure.

```
droelf# ./pcap_extractor -i 0/vm_transfer.pcap -o out_ennorey.pcap -f "host 192.168.0.13" -s "Enno Rey"
pcap highspeed extractor version 1       by Daniel Mende - dmende@ernw.de
Found 5 matching packets in 103 seconds.
droelf#
```

Figure 2: Console output of pcap_extractor

The resulting output file contains the packets with the data searched for, as shown below.

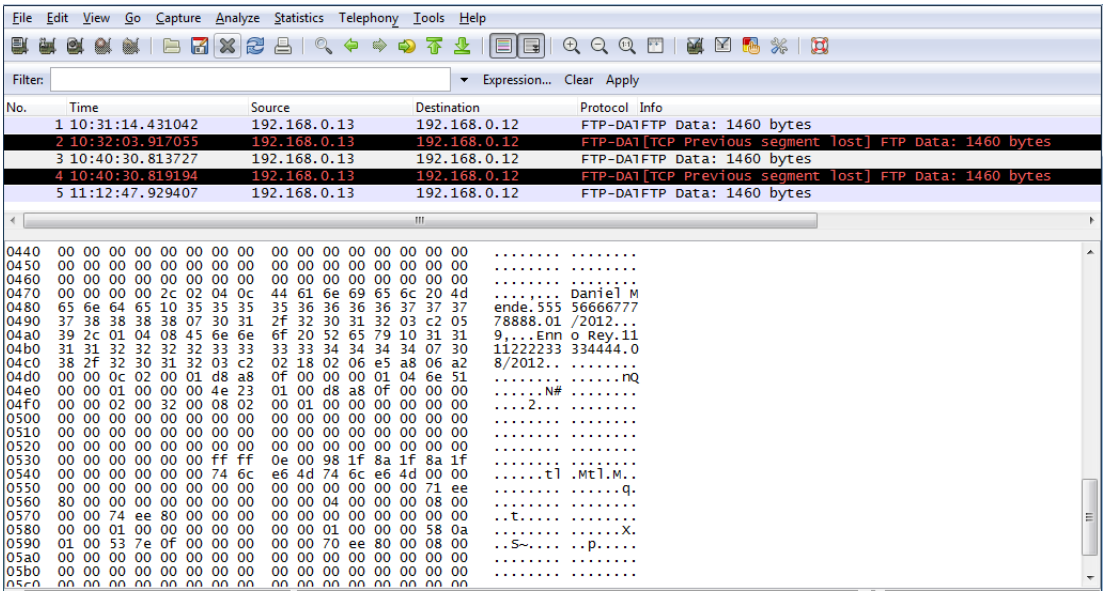


Figure 3: Outfile of pcap_extractor in Wireshark