From Full Capture to Criminal Evidence: A Real-World Case of Lawful Interception

Daniel Spiekermann

Overview

- About me
- Background
  - Lawful Interception
  - The case
  - About the complexity of the capture process
- Finding the needle in the needlestack
- Analysis
- Summary
- Lessons learned

About me

Daniel Spiekermann

Current position:

Professor for Distributed Systems at University of Applied Science and Arts in Dortmund, Germany (since 2023)

Previous positions:

Professor for Digital Investigations at Police Academy of Lower Saxony, Germany (2020 - 2023)

Digital Investigator at Police Headquarter Dortmund, Germany (2013 – 2020)

… and some other IT jobs (mainly network and security) prior

How it starts...

In January 2021, the department of public prosecution in Dortmund called me and asked a simple question

How it starts...

„Are you willing to analyse network packets of a lawful interception to reveal criminal activities?”

How it starts...

Definitely YES

Lawful Interception

- General term for police/state interception of telecoms
- Covers phone, VoIP, SMS, email, and internet traffic at the provider level
- Requires a judicial order, except in very limited urgent cases

Very, very, very simplified: Three relevant regulations

- §100a German Code of Criminal Procedure (Strafprozessordnung (StPO))
- Telecommunications Act - Telekommunikationsgesetz (TKG))
- Technical Guideline for the implementation of legal measures for the surveillance of telecommunications and the disclosure of information

Lawful Interception

- General term for police/state interception of telecoms
- Covers phone, VoIP, SMS, email, and internet traffic at the provider level
- Requires a judicial order, except in very limited urgent cases

Very, very, very simplified: Three relevant regulations

- §100a German Code of Criminal Procedure (Strafprozessordnung (StPO))
- Telecommunications Act - Telekommunikationsgesetz (TKG))
- Technical Guideline for the implementation of legal measures for the surveillance of telecommunications and the disclosure of information (Technische Richtlinie Telekommunikationsüberwachungsverordnung (TR TKÜV)

§100a StPO - German Code of Criminal Procedure

(1) Telecommunications may also be monitored and recorded without the knowledge of the persons concerned if:

1. Specific facts establish a suspicion that a person, as a perpetrator or participant, has committed, attempted to commit (in cases where the attempt is punishable), or prepared for a serious criminal offense specified in subsection (2),

2. The offense is serious even in the individual case, and

3. The investigation of the facts or the ascertainment of the whereabouts of the accused would otherwise be significantly hampered or hopeless.

[The following offenses are included, among others, from subsection (2):]

- Murder and Manslaughter pursuant …
- Dissemination, Procurement, and Possession of Child and Youth Pornographic Content pursuant …
- Fraud and Computer Fraud …

§100a StPO - German Code of Criminal Procedure

(1) **Telecommunications** may also be **monitored and recorded without the knowledge** of the persons concerned if:

1. Specific facts establish a suspicion that a person, as a perpetrator or participant, has committed, attempted to commit (in cases where the attempt is punishable), or prepared for a serious criminal offense specified in subsection (2),

2. The offense is serious even in the individual case, and

3. The **investigation of the facts** or the ascertainment of the whereabouts of the accused **would otherwise be significantly hampered or hopeless**.

[The following offenses are included, among others, from subsection (2):]

- Murder and Manslaughter pursuant …
- Dissemination, Procurement, and Possession of Child and Youth Pornographic Content pursuant …
- Fraud and **Computer Fraud** …

Telekommunikationsgesetz aka Telecommunications Act:

- Regulates obligations of telecom providers
- Providers must maintain technical interfaces for lawful interception and cooperate with law enforcement when presented with valid orders
- Specifies cost reimbursement rules for providers
- Technical standards are aligned with ETSI lawful interception specifications

Telecommunications Act

Telekommunikationsgesetz aka Telecommunications Act:

- Regulates obligations of telecom providers
- Providers must maintain **technical interfaces** for **lawful interception** and **cooperate with law enforcement** when presented with valid orders
- Specifies cost reimbursement rules for providers
- Technical standards are aligned with ETSI lawful interception specifications

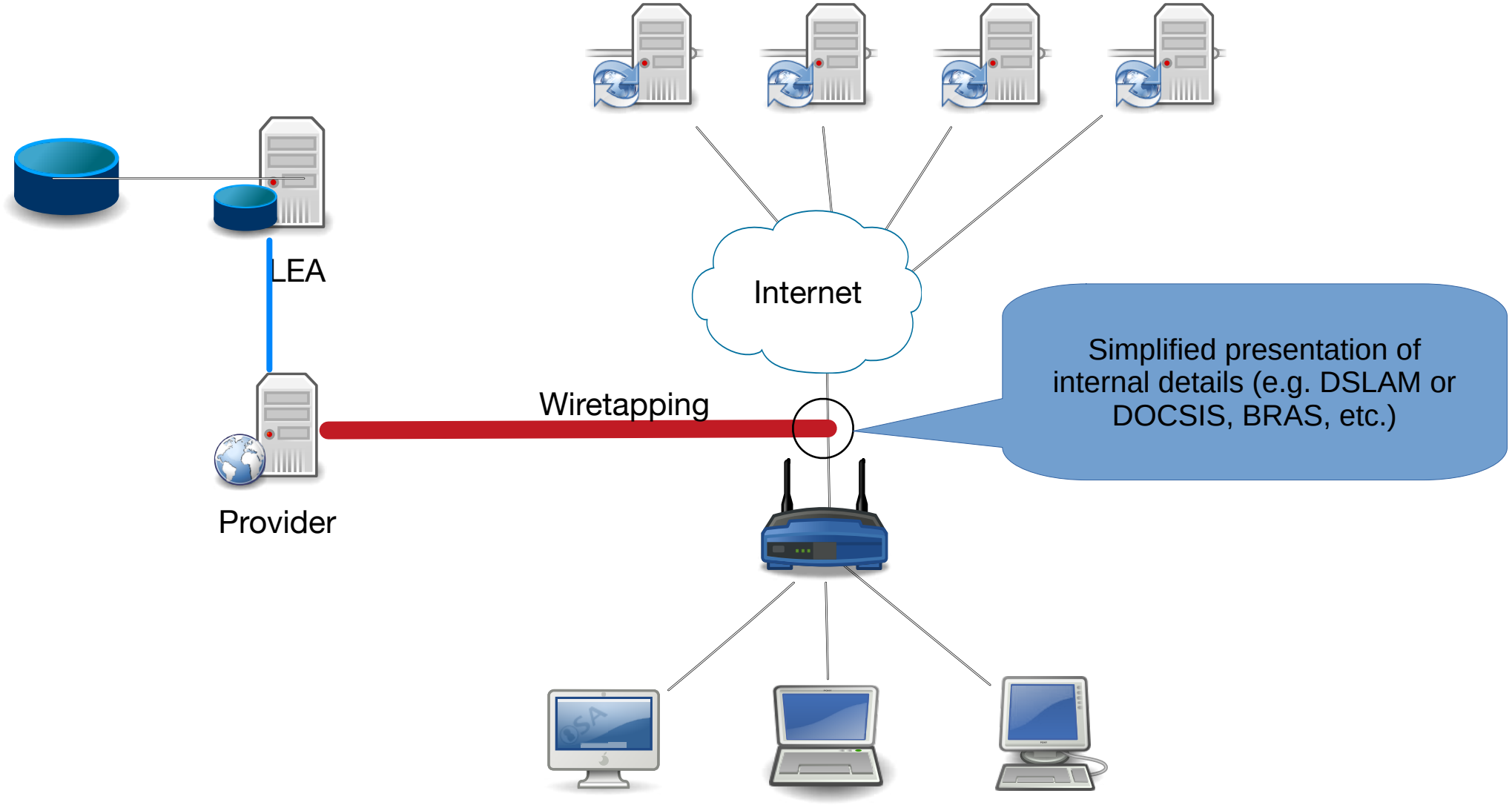
Bases on different ETSI documents (European Telecommunications Standards Institute)

- e.g. ETSI TS 101 671 V3.15.1 (2018-06), ETSI TS 102 232-1 V3.32.1 (2024-07), ETSI TS 104 007 V1.1.1 (2024-11)
- Providers do not just dump raw data
- Instead, ETSI defines three separate standardized handover interfaces:
  - HI1 – Administrative / warrant information
  - HI2 – Intercept Related Information (IRI, metadata)
  - HI3 – Content of Communication (CC, the payload)
    - The actual communication stream (voice, SMS, IP packets, VoIP payload, email content, etc.)
    - Must be delivered 1:1 and in near real time
    - Transported via secure IP tunnels (IPSec, TLS) in standardized formats (e.g., RTP-like for voice)

Bases on different ETSI documents (European Telecommunications Standards Institute)

- e.g. ETSI TS 101 671 V3.15.1 (2018-06), ETSI TS 102 232-1 V3.32.1 (2024-07), ETSI TS 104 007 V1.1.1 (2024-11)
- Providers do **not just dump raw data**
- Instead, ETSI defines three separate standardized handover interfaces:
  - HI1 – Administrative / warrant information
  - HI2 – Intercept Related Information (IRI, metadata)
  - **HI3 – Content of Communication** (CC, the payload)
    - The actual communication stream (voice, SMS, IP packets, VoIP payload, email content, etc.)
    - Must be **delivered 1:1 and in near real time**
    - Transported via secure IP tunnels (IPSec, TLS) in standardized formats (e.g., RTP-like for voice)

A lot of text, how is it done in reality



Disclaimer

As the captured packets and extracted data cover a real world case, some information are anonymized.

Most scripts are shortened.

Some script snipplets are simplified and may differ from the used one in 2021 (re-tested with Wireshark 4.6.0)

The case

- Pay-TV-Card-Sharing
  - Perpetrator shares access to encrypted pay-TV channels
  - Affected provider: Sky Germany
- Technical Implementation:
  - Manipulation of a receiver (customized firmware is installed)
  - Receiver requests session keys from a defined server in the internet (server is defined in the custom firmware)
  - Session keys facilitate decryption of the signal

The case in detail

- A business is illegally and under the table offering the possibility to provide a tampered receiver and charges a certain amount of money for it.
- With this manipulated receiver, the “customer” is able to watch the complete television programm of Sky Germany for a single payment (approx: 50 – 100 EUR)
  - Official price of Sky Germany (full packet) in 2021: 70 EUR p.m.
- LI focuses on the business, which is co-located in the house of the shop owner

The case in detail

- Capturing 3 cable-based (DOCSIS 3.1) internet access points:
  - 1 x 1000 Mbit/s
  - 2 x 500 Mbit/s

- The guidelines are quite simply for this
  - The provider has to transfer the data to the Law Enforcement Agency (LEA)
  - The LEA has to capture the data …

The case in detail

- Capturing 3 cable-based (DOCSIS 3.1) internet access points:
  - 1 x 1000 Mbit/s
  - 2 x 500 Mbit/s
- The guidelines are quite simply for this
  - The provider has to transfer the data to the Law Enforcement Agency (LEA)
  - The LEA has to capture the data …
- Unfortunately:
  - No german LEA was able to collect this amount of data without breaking the running LIs

Statistics - Phase 1

- 1 Gbit/s line wiretapped
  - Start: 11. February 2021 09:46:37 CET
  - End: 12. February 2021 15:52:49 CET

- 2030 snoop-files with 149.557.059 packets

Capture takes place at a state criminal investigation office in Germany

Statistics - Phase 2

- Both 500 Mbit/s lines wiretapped
  - Start: 30. March 2021 08:47:42 CEST
  - End: 06. April 2021 23:58:01 CEST

- Line 1: 13.865 snoop-files with 907.183.700 packets
- Line 2: 7.191 snoop-files with 536.741.681 packets

- Overall 1.443.925.381 packets

Capture takes place at the federal criminal police office in Germany

Step 1

Comparing the hash values of the snoop files with a list of hash values provided by the LEA

- Important step in digital investigations
- Ensures integrity of the data
- MD5 is ok for this
- Comparing 23086 files takes some time

```
md5 *.snoop
MD5 (00002_2021xxxxxx.snoop) = a1454235c962fe3794387915b4c7f1b9
MD5 (00003_2021xxxxxx.snoop) = 4ad5e722d7de4355ef4688edcd7fbeda
MD5 (00004_2021xxxxxx.snoop) = 6190cd9b90c257b1851ed7f17dd5e41d
MD5 (00005_2021xxxxxx.snoop) = 6275bd2287cda4223bf305a2938698f1
MD5 (00006_2021xxxxxx.snoop) = ab9cf6bb1d75bb83ad37dabaf79a1e36
MD5 (00007_2021xxxxxx.snoop) = 54758d09a272e8530634258151 33f66a
MD5 (00008_2021xxxxxx.snoop) = 5dbdfc2281c0829b1a9c3c5b5168a207
...
```

Converting

- Converting the snoop to pcap
  - Snoop is a okay, but some tools lack in supporting it

Length:	77 MB
Hash (SHA256):	7e183f801cdd177afc716ee0d64341738014a6e26d94385e38aeef6f9a1a0a8e
Hash (SHA1):	66d6409f4662094c8a0e32ef7dd230a6b7ae25f3
Format:	Sun snoop
Encapsulation:	Ethernet
Time	
First packet:	2021-02-12 14:11:55
Last packet:	2021-02-12 14:12:38
Elapsed:	00:00:42
Capture	
Hardware:	Unknown
OS:	Unknown
Application:	Unknown

  - Easiest way to convert:

    `tshark -r IN.snoop -w OUT.pcap`
  - Use of bash for scripting (and again md5 calculating)

```
for i in *.snoop;
do
        tshark -r $i -w ../raw_pcap_format/$i.pcap;
        md5 ../raw_pcap_format/$i.pcap >> hashvalue.txt
done
```

Start the search…

How can you start without any clues or starting points?

Start the search…

- How can you start without any clues or starting points?

- Getting an overview with Wireshark / tshark
  - Statistics -> Conversations
  - Statistics -> Endpoints
  - Statistics -> Protocol hierarchy

- `tshark -r pcap1.pcap -z conv,tcp`
- `tshark -r pcap1.pcap -z conv,ip`
- `tshark -r pcap1.pcap -z conv,udp`

Start the search…

- How can you start without any clues or starting points?

- Getting an overview with Wireshark / tshark
  - Statistics -> Conversations
  - Statistics -> Endpoints
  - Statistics -> Protocol hierarchy

Not quite useful with > 20000 files

- `tshark -r pcap1.pcap -z conv,tcp`
- `tshark -r pcap1.pcap -z conv,ip`
- `tshark -r pcap1.pcap -z conv,udp`

Overview

- No usable / interesting protocols or endpoints on first sight
- But some endpoints have a high throughput (= TopTalker)

Address A	Port A	Address B	Port B	Packets	Bytes	Bytes A → B	Bytes B → A
[redacted]	52123	[redacted]	9001	15,448	22M	29k	22M
	59334		1080	19,005	19M	409k	19M
	33034		1087	15,709	16M	339k	15M
	37750		1085	8,958	8893k	208k	8684k

Short reminder:
Snoop file captures 77MB
Capture file duration ~ 15 sec

TopTalker

- Detecting 3 (maybe) interesting systems
  - 5.9.73.xx
  - 136.243.133.xx
  - 116.202.247.xx
  - Checking communication of these three systems
    1) Check port (9001, 1080, 1085, 1087)

9001 (might be) Tor
1080 (might be) socks
1085 (might be) WebObjects
1087 (might be) CPL Scrambler Internal

```
└─$ cat /etc/services | grep 1080
socks           1080/udp    # Socks
socks           1080/tcp    # Socks
└─$ cat /etc/services | grep 1085
webobjects 1085/udp    # Web Objects
webobjects 1085/tcp    # Web Objects
└─$ cat /etc/services | grep 1087
cplscrambler-in 1087/udp    # CPL Scrambler Internal
cplscrambler-in 1087/tcp    # CPL Scrambler Internal
```

Detecting 3 interesting systems

- 5.9.73.xx
- 136.243.133.xx
- 116.202.247.xx

- Checking communication of these 3 systems
  1) Check port (9001, 1080, 1085, 1087)
  2) Search 3-Way Handshake (simplified via tcp.flags.syn==1)
  3) Check the subsequent packets for relevant / readable / usable information like this:

```
50 e6 36 d7 f3 7c ba a6  e3 91 f6 3a 08 00 45 4a   P·6··|·· ···:··EJ
00 4a 00 00 00 00 40 06  fc e9 c0 a8 0a 7d 5b fa   ·J····@· ·····}[·
56 5b c7 a8 57 3e e8 fc  eb b8 e5 c9 7f 13 80 18   V[··W>·· ········
08 16 cf e1 00 00 01 01  08 0a 92 f9 1d 0e 74 50   ········ ·····tP
e9 82 53 53 48 2d 32 2e  30 2d 4f 70 65 6e 53 53   ··SSH-2. 0-OpenSS
48 5f 31 30 2e 30 0d 0a                           H_10.0··
```

TopTalker

- Not interesting on first sight, so filter the top talker out of the data

- Create new pcaps without any of these ip-addresses

```
for i in *.pcap;
do
    tshark -r $i -Y !ip.addr == 5.9.73.x -w $i_filtered.pcap
done
```

Top Talker

- Amount of data reduced (approx 60%), but still lots of packets (> 600,000,000 packets)
- I should remove packets I do not need ...
- How to remove packets of irrelevant communication?

Top Talker

- Amount of data reduced (approx 60%), but still lots of packets
- I should remove packets I do not need …
- How to remove packets of irrelevant communication?
- What is irrelevant communication?

Top Talker

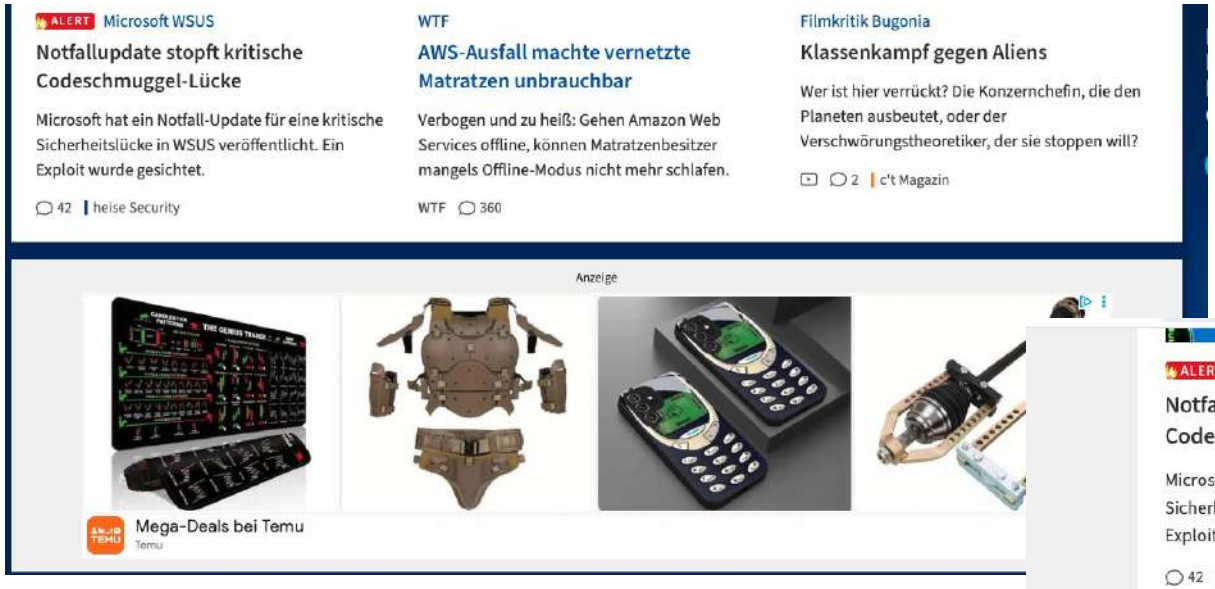
- Amount of data reduced (approx 60%), but still lots of packets
- I should remove packets I do not need …
- How to remove packets of irrelevant communication?
- What is irrelevant communication?
- Everything that is not related to the case

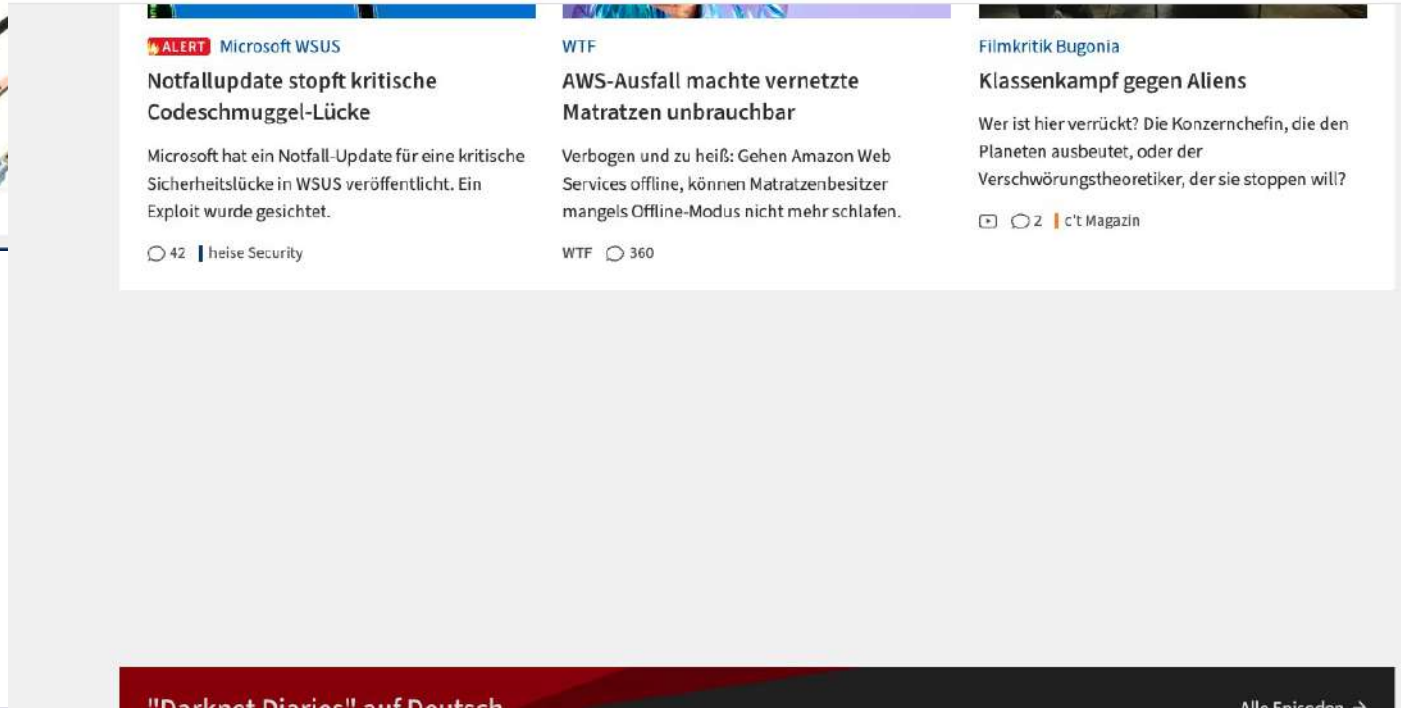
Magic of DNS

- The network packets are a result of communication between two systems
- One system is inside the wiretapped network
- The orther system might be requestet from this system
  - by IP
  - by name
- DNS is the system to manage these information
- Use DNS to determine IP <-> Name mapping
- Find FQDNs that might be irrelevant
  - Educated guess: google, youtube, etc.

Determine mostly irrelevant systems

- Everyone knows irrelevant / annoying parts of websites and services ...





DNS information

```
for i in *.pcap;
do
  tshark -r $i -Y dns -T fields -e dns.resp.name -e dns.a >> ../analysis/dns_req_resp.txt;
done
```

Example result:

pull-hls-q5-va01.tiktokcdn.com,pull-hls-q5-

va01.tiktokcdn.com.akamaized.net,a1852.z.akamai.net,a1852.z.

akamai.net 95.101.134.152,95.101.134.209

web.facebook.com,star.facebook.com,star.c10r.facebook.com
179.60.192.3

footprints-pa.googleapis.com 216.58.213.74

play.googleapis.com 142.250.74.234

android.googleapis.com | *216.58.215.42*

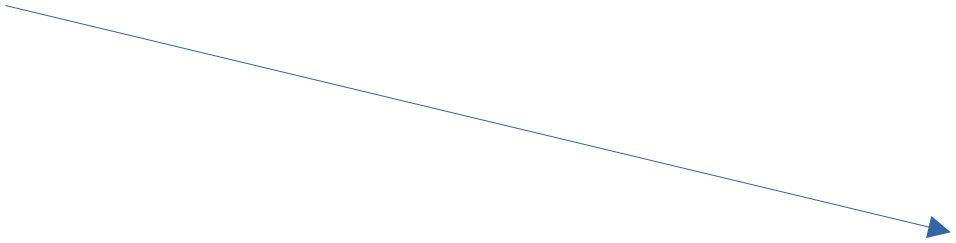
dns.resp.name

dns.a

Determine mostly irrelevant systems

Taking a list of known ad and tracking sites

- https://github.com/mrxehmad/pi-hole-adlist
- https://github.com/topics/pihole-blocklists
- https://github.com/hagezi/dns-blocklists
- https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts



```
====================================
# Title: Hosts contributed by Steven Black
# http://stevenblack.com

0.0.0.0 ad-assets.futurecdn.net
0.0.0.0 ck.getcookiestxt.com
0.0.0.0 eu1.clevertap-prod.com
0.0.0.0 wizhumpgyros.com
0.0.0.0 coccyxwickimp.com
0.0.0.0 webmail-who-int.000webhostapp.com
0.0.0.0 010sec.com
0.0.0.0 01mspmd5yalky8.com
0.0.0.0 0byv9mgbn0.com
0.0.0.0 ns6.0pendns.org
0.0.0.0 dns.0pengl.com
0.0.0.0 12724.xyz
```

A little bit of bash-foo, and the list was exported in a new file containing only the FQDN

Added some adult sites and

Alexa Top 100 (EOL 2022)

DNS find, extract, unique and sort


```
for i in *.pcap
    do
    for n in $(cat names.txt)
    do
        if tshark -r $i -Y "dns.resp.name contains $n" -T fields -e dns.a | grep -q .
         then echo "$n: $(tshark -r $i -Y "dns.resp.name==$n" -T fields -e dns.a |
        sort -u | xargs)" >> found_ips.txt
        fi
    done
done
```

```
9,2.23.13.156,2.23.13.160 95.100.95.168,95.100.95.185,95.100.95.172,95.100.95.166,95.100.95.132,95.100.95.191,95.100.95.174,95.100.95.137,95.100.95.133
95.100.95.177,95.100.95.161,95.100.95.180,95.100.95.155,95.100.95.163,95.100.95.181,95.100.95.146,95.100.95.183,95.100.95.187
95.100.95.191,95.100.95.185,95.100.95.168,95.100.95.133,95.100.95.162,95.100.95.132,95.100.95.172,95.100.95.174,95.100.95.137
88.221.198.44,88.221.198.7,88.221.198.25,88.221.198.23,88.221.198.58,88.221.198.29,88.221.198.50,88.221.198.52,88.221.198.51
95.100.252.57,95.101.134.168 95.101.134.152,95.101.134.209 95.101.134.162,95.101.134.232 95.101.134.168,95.101.134.161 95.101.134.179,95.101.134.185
95.101.134.186,95.101.134.225 95.101.134.195,95.101.134.178 95.101.134.203,95.101.134.155 95.101.134.209,95.101.134.152 95.101.134.217,95.101.134.178
95.101.134.219,95.100.252.59 95.101.134.226,95.101.134.154 95.101.134.226,95.101.134.168 95.101.134.227,95.101.134.184 95.101.134.232,95.101.134.163
95.101.134.232,95.101.134.194
```

DNS find, extract, unique and sort pt. 2

```
cat found_ips.txt| tr '\n ' ', ' | sort -u >> ips.txt
```

```
187.15,23.90.187.16,10
4.166.129.4,104.166.12
9.5,23.90.187.4,23.90.
187.12,23.90.187.5,23.
90.187.9,23.90.187.10,
23.90.187.8,23.90.187.
9,23.90.187.6,23.90.18
7.16,104.166.12
```

csv-list contains approx 7500 IPs (only IPv4)

```
for i in *.pcap
do
        tshark -r $i -Y "not (ip.addr in {$(<ips.txt)})" -w filtered_$i.pcap
done
```

Amount of data reduced (now again approx 65%, remaining > 280,000,000)

Results

- Much better, but took very long
  - approx 47 hours
  - MacBook Air 2020
    - 2,0 GHz QuadCore Intel Core i5 (10. Generation)
    - 32 GB RAM

- Still not possible to analyze the data manually (approx. 280 million packets)

Filtering part 3

Remove every protocol which is typically encrypted or used for encryption, e.g.

- TLS on 443/TCP
- Wireguard on 51820/UDP
- Openvpn on 1194/UDP

But not

- SSH on 22/TCP

Might be used for administration

Filtering part 3

Amount of data further reduced (now approx 90 %)
At least more than 20 million packets

Filtering part 4

20 million packets is still too much

Sun Tzu*:
To know your enemy, you must become your enemy.

*Sun Tzu was a Chinese military general, strategist, philosopher, and writer who lived during the Eastern Zhou period (771–256 BC)

Think like a criminal

Or: Think like a detective

- Structured process for investigations based on the

5 Ws and 1 H

1) Who?
2) What?
3) Where?
4) When?
5) Why?
6) How?

Think like a criminal

Or: Think like a detective

- Structured process for investigations based on the

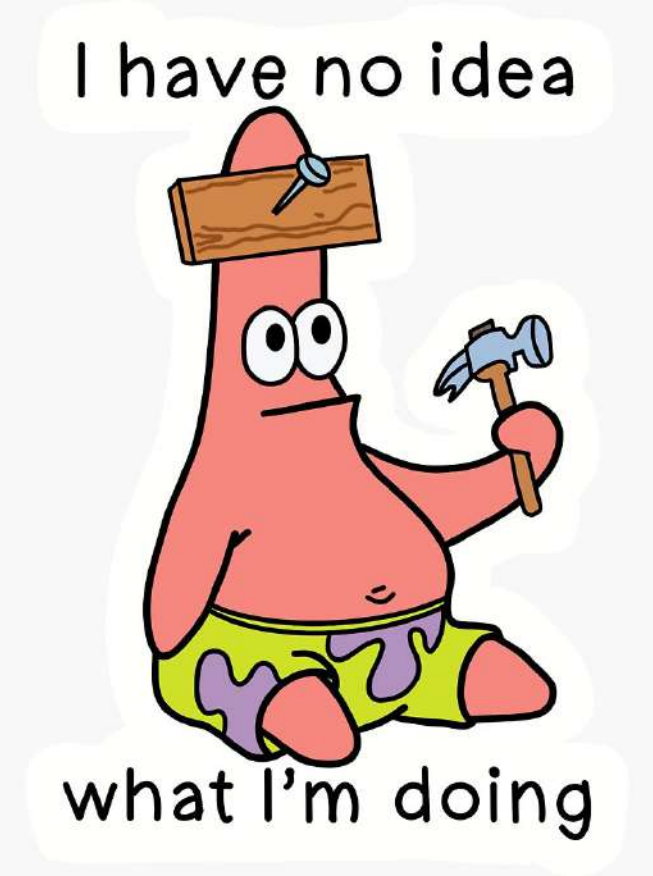
5 Ws and 1 H

1) Who?
2) What?
3) Where?
4) **When?**
5) Why?
6) **How?**

Think like a criminal

How would I do it?

- I have absolutely no idea



Think like a criminal

How would I do it?

- I have absolutely no idea

When would I do it?

- During office hours (9 am - 5 pm)
- As a night shift worker (12 am - 5am)

Well-known practice in attribution of attacks: Compare the working time with time zones all over the earth

Filtering part 4

Filtering for the time when something happens

First focus on office hours:

- Maybe people come into the shop and ask for: „Optimized provision of TV experience“

```
for i in *.pcap
do
       tshark -r $i -Y "frame.time_utc >= 2021-02-11T09:00:00.000000000Z” -w t_$i
done
```

Finally, the filtering based on office hours reduced the amount to approx. 10 million packets

Filtering part 5

Back to „How would I do it“?

If I would have to manage such service, I would prefer reliable communication, i.e. no UDP

Filtering out UDP significantly reduced the amount of remaining packets to at least 800.000.

Lets dive into a capture file

The first hit

60762	2021-02-11	11:50:52.037197	159.69…	109.91…	TCP	236 48044 → 3337 [PSH, ACK] Seq=6121 Ack=1513 Win=…
60763	2021-02-11	11:50:52.037846	109.91…	178.33…	TCP	66 43160 → 19000 [ACK] Seq=27 Ack=27 Win=913 Len=…
60768	2021-02-11	11:50:52.052915	109.91…	159.69…	TCP	66 3337 → 48044 [ACK] Seq=1513 Ack=6291 Win=659 L…
61185	2021-02-11	11:50:52.386328	109.91…	159.69…	TCP	108 3337 → 48044 [PSH, ACK] Seq=1513 Ack=6291 Win=…
61204	2021-02-11	11:50:52.395562	159.69…	109.91…	TCP	66 48044 → 3337 [ACK] Seq=6291 Ack=1555 Win=502 L…
61290	2021-02-11	11:50:52.416814	159.69…	109.91…	TCP	236 48044 → 3337 [PSH, ACK] Seq=6291 Ack=1555 Win=…
61312	2021-02-11	11:50:52.423580	109.91…	178.33…	TCP	66 53931 → 808 [SYN] Seq=0 Win=8192 Len=0 MSS=146…
61322	2021-02-11	11:50:52.433228	109.91…	159.69…	TCP	66 3337 → 48044 [ACK] Seq=1555 Ack=6461 Win=659 L…
61325	2021-02-11	11:50:52.437392	178.33…	109.91…	TCP	66 808 → 53931 [SYN, ACK] Seq=0 Ack=1 Win=14600 L…
61329	2021-02-11	11:50:52.453069	109.91…	178.33…	TCP	54 53931 → 808 [ACK] Seq=1 Ack=1 Win=65700 Len=0
61679	2021-02-11	11:50:52.675583	178.33…	109.91…	TCP	188 41283 → 3336 [PSH, ACK] Seq=977 Ack=337 Win=11…
61698	2021-02-11	11:50:52.680177	109.91…	178.33…	TCP	66 3336 → 41283 [ACK] Seq=337 Ack=1099 Win=361 Le…
61770	2021-02-11	11:50:52.763565	109.91…	159.69…	TCP	108 3337 → 48044 [PSH, ACK] Seq=1555 Ack=6461 Win=…
61785	2021-02-11	11:50:52.814267	159.69…	109.91…	TCP	66 48044 → 3337 [ACK] Seq=6461 Ack=1597 Win=502 L…
61791	2021-02-11	11:50:52.839786	159.69…	109.91…	TCP	188 55070 → 3336 [PSH, ACK] Seq=2319 Ack=799 Win=5…
61795	2021-02-11	11:50:52.852876	109.91…	159.69…	TCP	66 3336 → 55070 [ACK] Seq=799 Ack=2441 Win=227 Le…
62009	2021-02-11	11:50:52.915120	109.91…	178.33…	TCP	456 53931 → 808 [PSH, ACK] Seq=1 Ack=1 Win=65700 L…
62041	2021-02-11	11:50:52.929106	178.33…	109.91…	TCP	54 808 → 53931 [ACK] Seq=1 Ack=403 Win=15744 Len=…
62082	2021-02-11	11:50:52.940397	178.33…	109.91…	TCP, HiP…	85 808 → 53931 [PSH, ACK] Seq=1 Ack=403 Win=15744

Not fair, but I disabled the coloring rules for this ;-)

The first hit

60762	2021-02-11	11:50:52.037197	159.69…	109.91…	TCP	236 48044 → 3337 [PSH, ACK] Seq=6121 Ack=1513 Win=…
60763	2021-02-11	11:50:52.037846	109.91…	178.33…	TCP	66 43160 → 19000 [ACK] Seq=27 Ack=27 Win=913 Len=0
60768	2021-02-11	11:50:52.052915	109.91…	159.69…	TCP	66 3337 → 48044 [ACK] Seq=1513 Ack=6291 Win=659 Le…
61185	2021-02-11	11:50:52.386328	109.91…	159.69…	TCP	108 3337 → 48044 [PSH, ACK] Seq=1513 Ack=6291 Win=…
61204	2021-02-11	11:50:52.395562	159.69…	109.91…	TCP	66 48044 → 3337 [ACK] Seq=6291 Ack=1555 Win=502 Le…
61290	2021-02-11	11:50:52.416814	159.69…	109.91…	TCP	236 48044 → 3337 [PSH, ACK] Seq=6291 Ack=1555 Win=…
61312	2021-02-11	11:50:52.423580	109.91…	178.33…	TCP	66 53931 → 808 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
61322	2021-02-11	11:50:52.433228	109.91…	159.69…	TCP	66 3337 → 48044 [ACK] Seq=1555 Ack=6461 Win=659 Le…
61325	2021-02-11	11:50:52.437392	178.33…	109.91…	TCP	66 808 → 53931 [SYN, ACK] Seq=0 Ack=1 Win=14600 Le…
61329	2021-02-11	11:50:52.453069	109.91…	178.33…	TCP	54 53931 → 808 [ACK] Seq=1 Ack=1 Win=65700 Len=0
61679	2021-02-11	11:50:52.675583	178.33…	109.91…	TCP	188 41283 → 3336 [PSH, ACK] Seq=977 Ack=337 Win=11…
61698	2021-02-11	11:50:52.680177	109.91…	178.33…	TCP	66 3336 → 41283 [ACK] Seq=337 Ack=1099 Win=361 Len…
61770	2021-02-11	11:50:52.763565	109.91…	159.69…	TCP	108 3337 → 48044 [PSH, ACK] Seq=1555 Ack=6461 Win=…
61785	2021-02-11	11:50:52.814267	159.69…	109.91…	TCP	66 48044 → 3337 [ACK] Seq=6461 Ack=1597 Win=502 Le…
61791	2021-02-11	11:50:52.839786	159.69…	109.91…	TCP	188 55070 → 3336 [PSH, ACK] Seq=2319 Ack=799 Win=5…
61795	2021-02-11	11:50:52.852876	109.91…	159.69…	TCP	66 3336 → 55070 [ACK] Seq=799 Ack=2441 Win=227 Len…
62009	2021-02-11	11:50:52.915120	109.91…	178.33…	TCP	456 53931 → 808 [PSH, ACK] Seq=1 Ack=1 Win=65700 Le…
62041	2021-02-11	11:50:52.929106	178.33…	109.91…	TCP	54 808 → 53931 [ACK] Seq=1 Ack=403 Win=15744 Len=0
62082	2021-02-11	11:50:52.940397	178.33…	109.91…	TCP, HiP…	85 808 → 53931 [PSH, ACK] Seq=1 Ack=403 Win=15744

Follow -> TCP-Stream


```
GET /CCcams7/images/users.png HTTP/1.1
Host: ya[redacted]dns.net:808
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: image/webp,*/*
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://ya[redacted]dns.net:808/CCcams7/
DNT: 1
Connection: keep-alive
Cookie: testing=1; sid=2adb58cfcf6881fee39cc0c89ce5cc10


HTTP/1.0 200 Document follows
Date: Thu, 11 Feb 2021 14:28:01 GMT
Server: MiniServ/1.590
Content-type: image/png
Last-Modified: Mon, 9 Jul 2012 15:56:23 GMT
Expires: Thu, 18 Feb 2021 14:28:04 GMT
Content-length: 3976
Connection: Keep-Alive

.PNG
.
...
IHDR...@...@.....tEXtSoftware.Adobe ImageReadyq.e<....PLTE.....m....f.......HHH..O.....
.s.......988.....t..Q.....Y.....ccc......kjj.....{..6SS
...\\\.....m...(((......zTz....xL.....9....Z.....lB..+.....u..d..R...fG5....pO..
```

Follow -> TCP-Stream


```
GET /CCcams7/images/users.png HTTP/1.1
Host: yan[redacted]s.net:808
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: image/webp,*/*
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://y[redacted]dns.net:808/CCcams7/
DNT: 1
Connection: keep-alive
Cookie: testing=1; sid=2adb58cfcf6881fee39cc0c89ce5cc


HTTP/1.0 200 Document follows
Date: Thu, 11 Feb 2021 14:28:01 GMT
Server: MiniServ/1.590
Content-type: image/png
Last-Modified: Mon, 9 Jul 2012 15:56:23 GMT
Expires: Thu, 18 Feb 2021 14:28:04 GMT
Content-length: 3976
Connection: Keep-Alive

.PNG
.
...
IHDR...@...@.....tEXtSoftware.Adobe ImageReadyq.e<....PLTE.....m....f.......HHH..O.....
.s.......988.....t..Q.....Y.....ccc......kjj.....{..6SS
...\\\.....m...(((......zTz....xL.....9....Z.....lB..+.....u..d..R...fG5....pO..
```

CCcam is a softcam (software emulator) for Linux-based satellite receivers that facilitates card sharing, a method of sharing a single subscription card over a network to decrypt paid television channels for multiple users.

Summary

- Looks like HTTP-communication
- But using port 808 (typically uncommon, but seen in some cybercrime investigation)

How can I make Wireshark dissect port 808 as HTTP?

Decode as

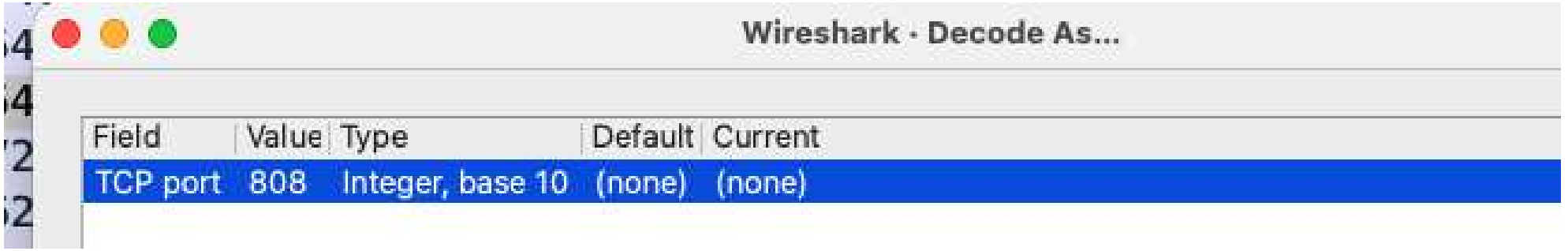
11.4.2. User Specified Decodes

The "Decode As" functionality lets you override what protocol is called under specific circumstances. This might be useful if Wireshark is incorrectly choosing which dissector to use for a particular TCP port, for example, or if you do some uncommon experiments on your network.

Decode as

11.4.2. User Specified Decodes

The "Decode As" functionality lets you override what protocol is called under specific circumstances. This might be useful if Wireshark is incorrectly choosing which dissector to use for a particular TCP port, for example, or if you do some uncommon experiments on your network.

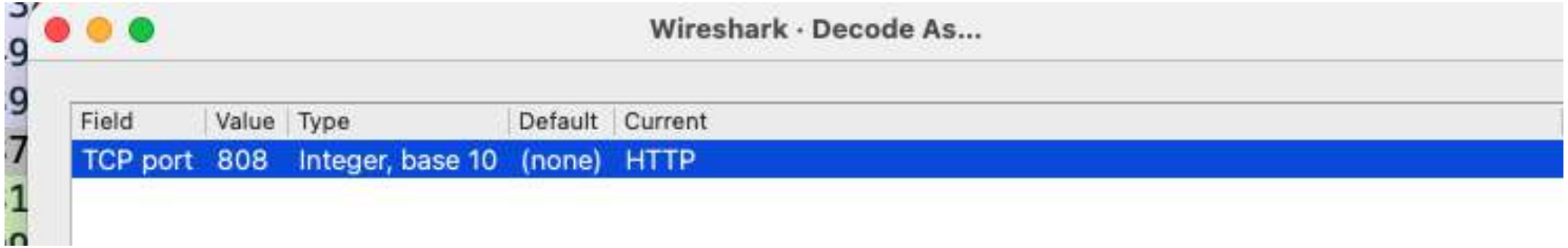


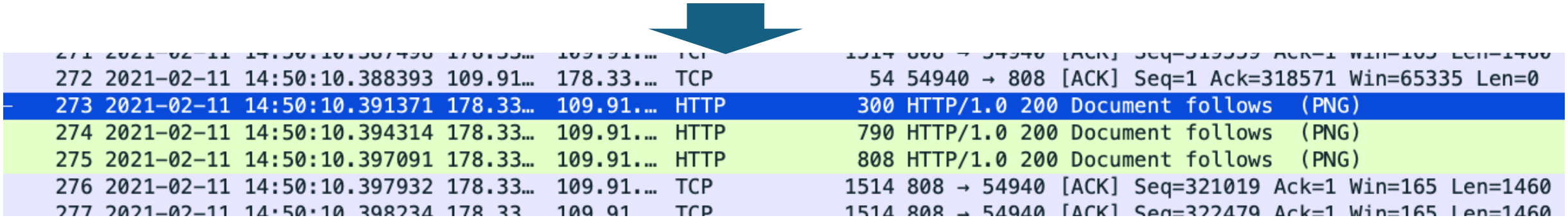

```
265 2021-02-11 14:50:10.378543 178.33… 109.91.… TCP  1514 808 → 54940 [ACK] Seq=315651 Ack=1 Win=165 Len=1460
266 2021-02-11 14:50:10.378546 178.33… 109.91.… TCP  1514 808 → 54940 [ACK] Seq=317111 Ack=1 Win=165 Len=1460
267 2021-02-11 14:50:10.378729 109.91… 178.33.… TCP    54 54940 → 808 [ACK] Seq=1 Ack=305431 Win=65335 Len=0
268 2021-02-11 14:50:10.380620 109.91… 178.33.… TCP    54 54937 → 808 [ACK] Seq=414 Ack=224 Win=16369 Len=0
269 2021-02-11 14:50:10.383313 109.91… 178.33.… TCP    54 54939 → 808 [ACK] Seq=413 Ack=224 Win=16098 Len=0
270 2021-02-11 14:50:10.384734 178.33… 109.91.… TCP  1042 808 → 54940 [PSH, ACK] Seq=318571 Ack=1 Win=165 Len=
```

Decode as

11.4.2. User Specified Decodes

The "Decode As" functionality lets you override what protocol is called under specific circumstances. This might be useful if Wireshark is incorrectly choosing which dissector to use for a particular TCP port, for example, or if you do some uncommon experiments on your network.





Merge packets according to their ip-address

1. Now things are getting easier, aren‘t they

2. Get all packets from or to this ip-address

```
for i in *.pcap
     tshark -r $i -Y „ip.addr == 178.33.x.y“ -w ip_$i
```

3. Merge them with mergecap into one file (just for simplicity)

4. Analyze it

Displayfilter http:

Protocol	Length	Info
HTTP	467	GET /CCcams3/images/download.png HTTP/1.1
HTTP	465	GET /CCcams3/images/enable.png HTTP/1.1
HTTP	466	GET /CCcams3/images/disable.png HTTP/1.1
HTTP	300	HTTP/1.0 200 Document follows (PNG)
HTTP	790	HTTP/1.0 200 Document follows (PNG)
HTTP	808	HTTP/1.0 200 Document follows (PNG)
HTTP	74	HTTP/1.0 200 Document follows (text/html)
HTTP	529	GET /CCcams2/ HTTP/1.1
HTTP	459	GET /CCcams2/images/allmanual.png HTTP/1.1
HTTP	455	GET /CCcams2/images/users.png HTTP/1.1
HTTP	463	GET /CCcams2/images/users_deleted.png HTTP/1.1
HTTP	459	GET /CCcams2/images/users_end.png HTTP/1.1
HTTP	1136	HTTP/1.0 200 Document follows (PNG)

Next steps

If you have identified one system, you can find more systems

- Using Wireshark/tshark and the string search for „Cccams“

```
for i in *.pcap
do
  tshark -r $i -Y 'frame contains "CCcams"'
done
```

Next steps

If you have identified one system, you can find more systems

- Using Wireshark/tshark and the string search for „Cccams“

```
for i in *.pcap
do
  tshark -r $i -Y 'frame contains "CCcams"'
done
```

Found 3 more systems

- System 2: 78.47.xx.yy:48888
- System 3: 138.201.xx.yy:5858
- System 4: 13.81.xx.yy:8880

Next step, analyze the server in detail

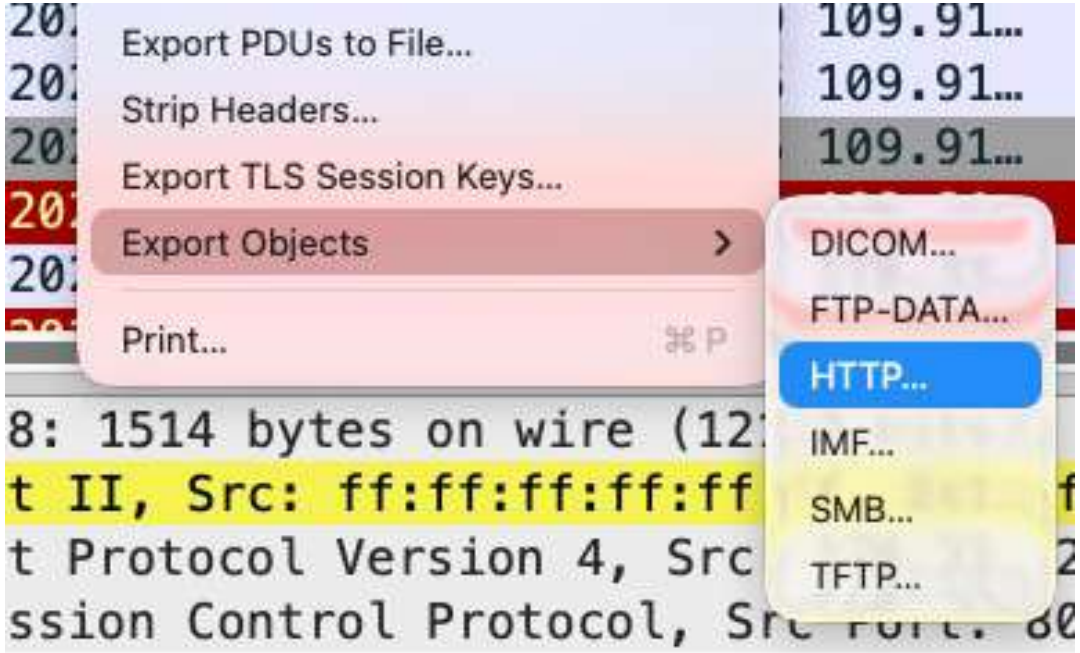
Knowing the involved systems is good, but having more details and finding the evidence is better

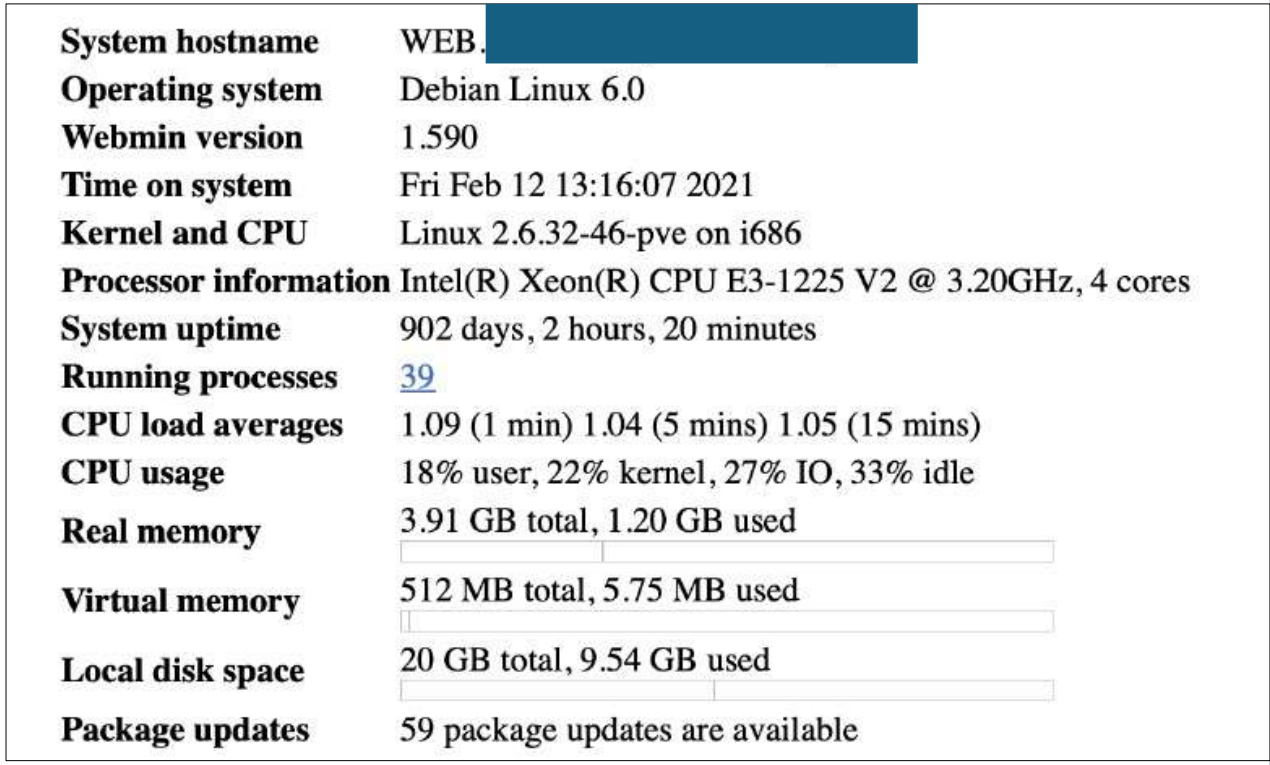
- The communication of port 808 is HTTP
- HTTP is plain text

Is it possible to reconstruct an entire communication?

Yes, it is

- Select packet -> Follow HTTP-stream -> store content in file
- File -> Export objects -> HTTP

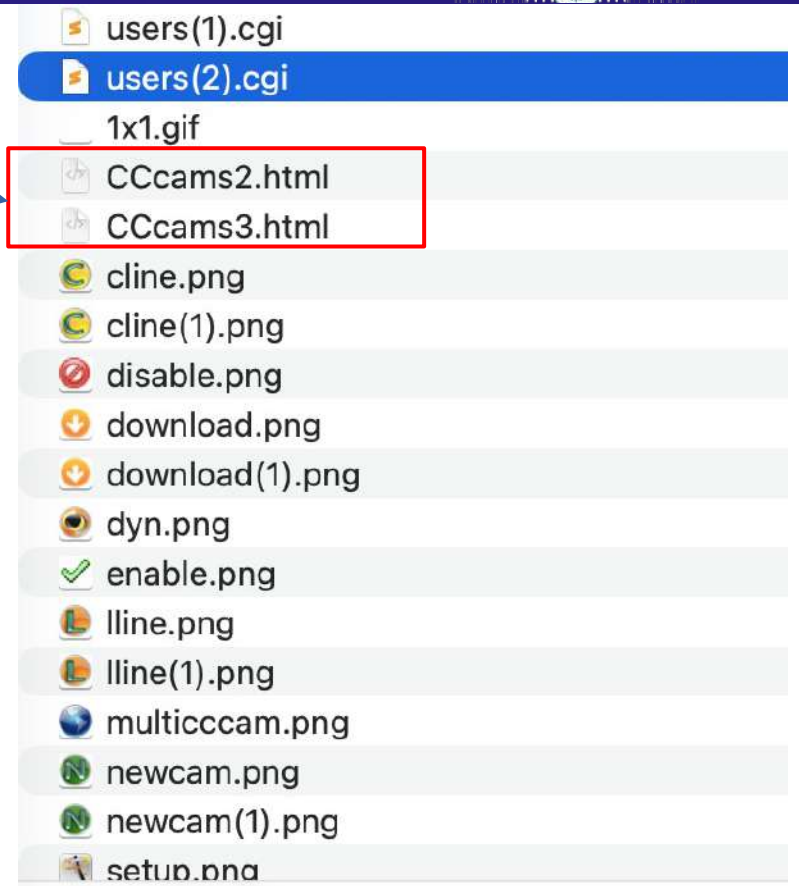




System hostname	WEB.
Operating system	Debian Linux 6.0
Webmin version	1.590
Time on system	Fri Feb 12 13:16:07 2021
Kernel and CPU	Linux 2.6.32-46-pve on i686
Processor information	Intel(R) Xeon(R) CPU E3-1225 V2 @ 3.20GHz, 4 cores
System uptime	902 days, 2 hours, 20 minutes
Running processes	39
CPU load averages	1.09 (1 min) 1.04 (5 mins) 1.05 (15 mins)
CPU usage	18% user, 22% kernel, 27% IO, 33% idle
Real memory	3.91 GB total, 1.20 GB used
Virtual memory	512 MB total, 5.75 MB used
Local disk space	20 GB total, 9.54 GB used
Package updates	59 package updates are available

Quick sidestep

- Extracts multiple (interesting) files like



- Gives a good overview, but
  - manual and repeated extraction over multiple files takes too much time
  - details about the „relevant“ connection is lost

Quick sidestep

I used tcpflow for reconstructing all http-traffic from the different pcaps (one pcap per server)

tcpflow(1) - Linux man page

Name

tcpflow - TCP flow recorder

Synopsis

tcpflow [**-chpsv**] [**-b** *max_bytes*] [**-d** *debug_level*] [**-f** *max_fds*] [**-i** *iface*] [**-r** *file*] [**expression**]

Description

tcpflow is a program that captures data transmitted as part of TCP connections (flows), and stores the data in a way that is convenient for protocol analysis or debugging. A program like ***tcpdump***(4) shows a summary of packets seen on the wire, but usually doesn't store the data that's actually being transmitted. In contrast, tcpflow reconstructs the actual data streams and stores each flow in a separate file for later analysis. tcpflow understands TCP sequence numbers and will correctly reconstruct data streams regardless of retransmissions or out-of-order delivery.

tcpflow stores all captured data in files that have names of the form

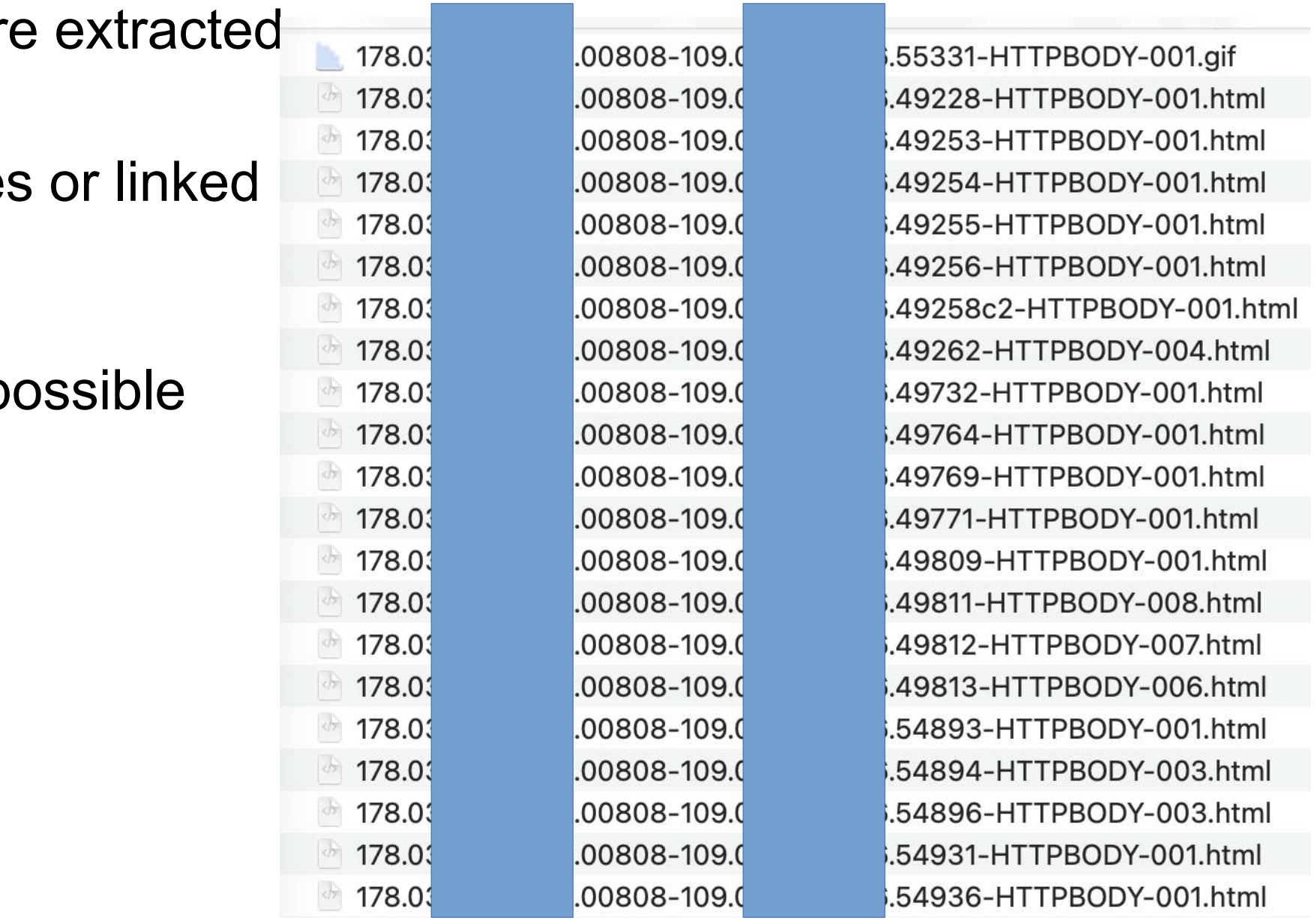
```
192.168.101.102.02345-010.011.012.013.45103
```

where the contents of the above file would be data transmitted from host 192.168.101.102 port 2345, to host 10.11.12.13 port 45103.

```
for i in *.pcap
    tcpflow -r $i -e all -o output
```

Quick sidestep

- Different flow were extracted
- Some with images or linked files
- Simple analysis possible



Extraced html-file

Modül Ayarları

cccam7-17000

Genel Firewall Ayarlar Araçlar Yeni Üye Ekle

Müşteri Listesi | CCcam Config | Silinmiş Müşteriler | Günü Dolmuş Müşteriler

CCcam info | Otomatik Kapatma | Auto Restart | CCcam Kontrol

1 Aylık Üyeler : 12
3 Aylık Üyeler : 6
6 Aylık Üyeler : 407
12 Aylık Üyeler : 12
Toplam Üyeler : 449
Aktif Üyeler : 245
Deaktif Üyeler : 204
Günü Dolmuş Üyeler : 204

CCcam Sharing Port : 5858
CCcam Web Info Port : 16006
CCcam Version : 2.0.11

CCcam Serveri Resetle
CCcam Serveri Durdur

Forensic investigation

First details about illegal provision of copyrighted „data“ found

Next steps:

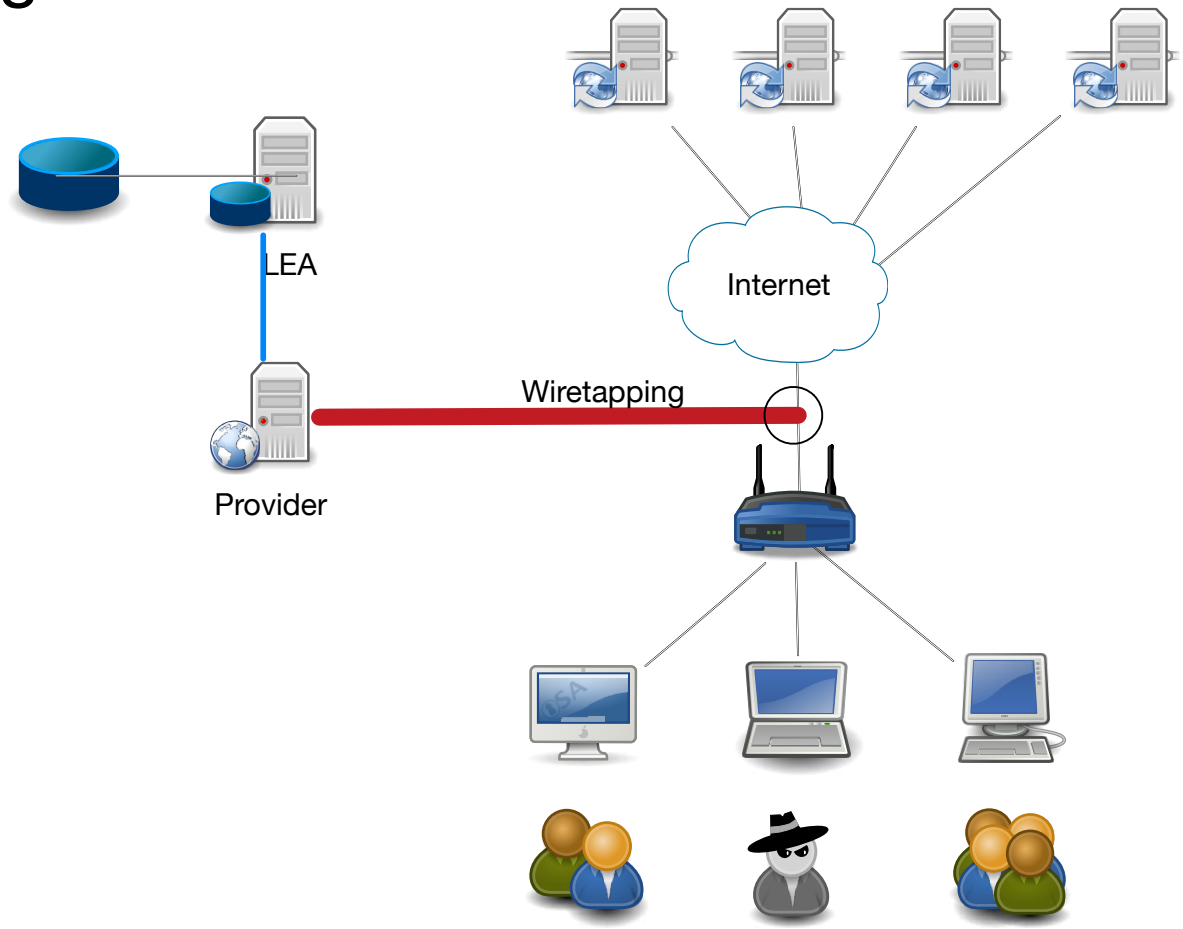
1. Find evidence about involved systems
2. Find evidence about perpetrator
3. Find details about „customers“

- Step 1 and 2 are necessary for reasoning of LEA and court case
- Step 3 might be relevant for further prosecution

Involved systems

How can I provide further details in case of a search?

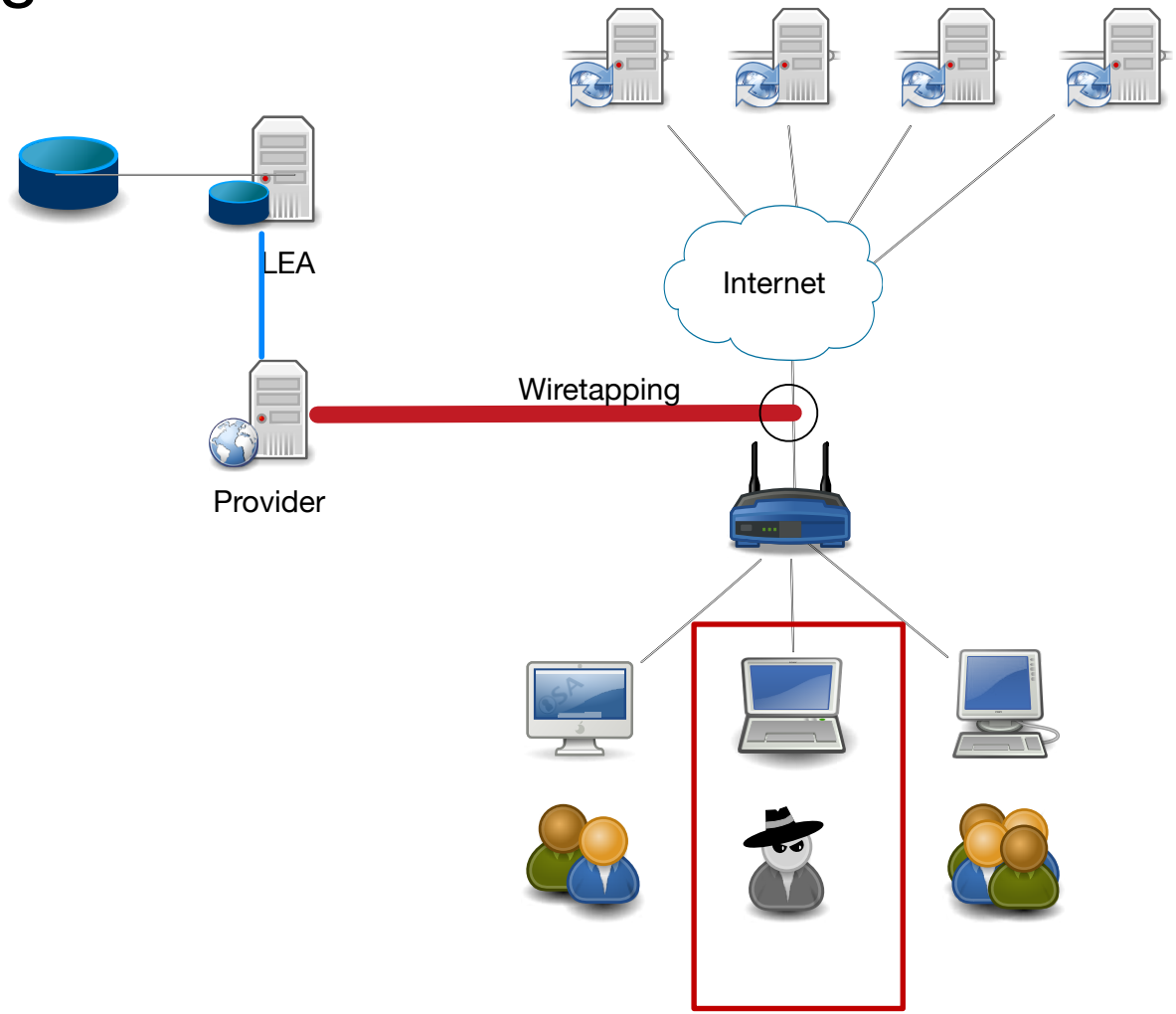
- What systems are used for the communication with the server?
- How can these systems be correctly identified?



Involved systems

How can I provide further details in case of a search?

- What systems are used for the communication with the server?
- How can these systems be correctly identified?



MAC-addresses

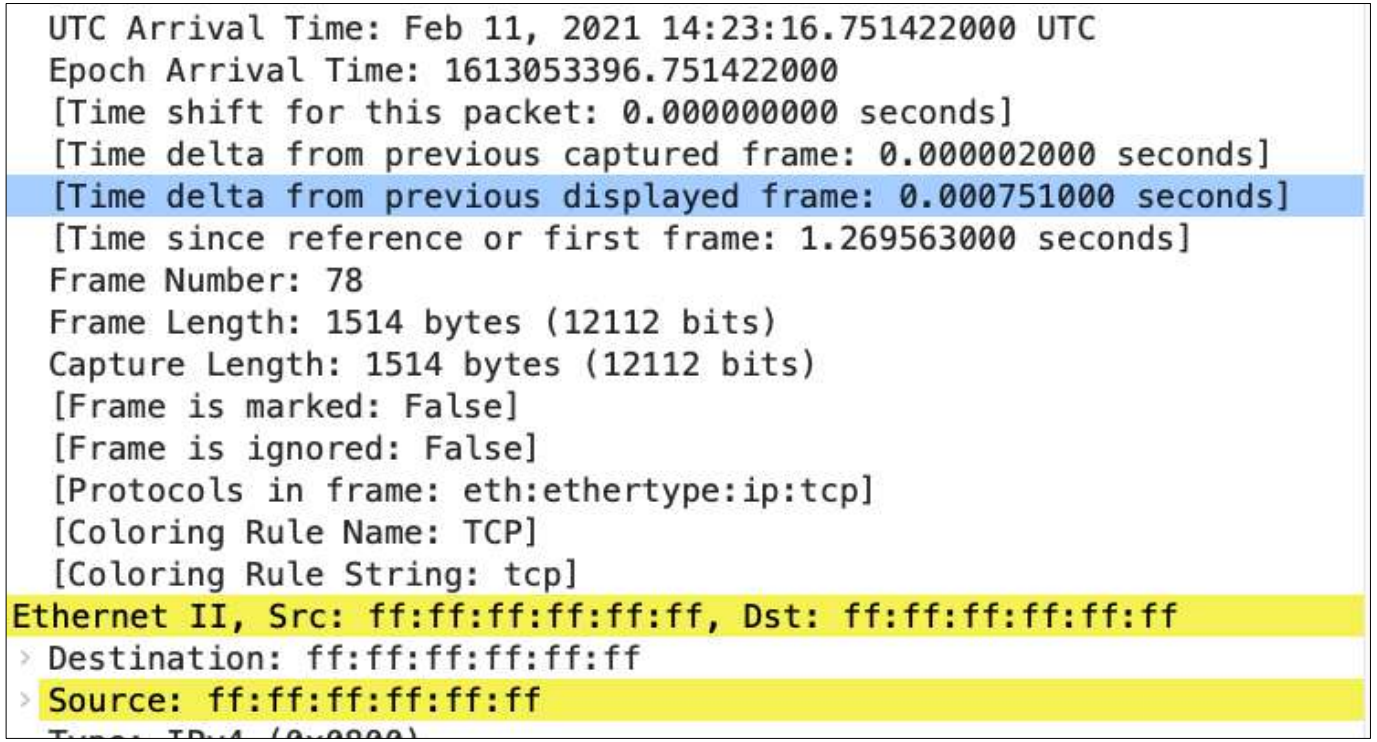
How can I provide data in case of a search?

- Unique identifiers like mac-addresses?

MAC-addresses

How can we provide data in case of a search?

- Unique identifiers like mac-addresses?
- Mac-addresses are not captured (because of capture position)



Forensic evidence 1 - user-agents

Extracting user-agent with tshark via:

```
-T fields -e http.user_agent
```

Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0

⇒ Seem to be a **Windows 7, with Firefox 85.0** system

Information about perpetrators

A common statement of perpetrators is:

„It wasn‘t me, my system was hacked“

So I have to find user-related information:

- Usernames
- Passwords

Information about perpetrators

There was a reconstructed web page with Webmin login …
Language seems to be turkish

(The shop owner is turkish)

How to find a username

Knowing details about Webmin and the plugin helps a lot…

Depending on the configuration of the webserver and Webmin, the process might differ, but typically you go to a specific URL and the a cgi-script named *session_login.cgi* is requested (via POST-method):

```
tshark -r server1.pcap -Y „http.request.method==POST &&
http.request.uri contains session_login“
```

```
68792  29.408089 109.91.xx.yy 178.33.xx.yy HTTP 635 POST
/session_login.cgi HTTP/1.1  (application/x-www-form-urlencoded)
```

Packet 67892


```
68792 2021-02-11 11:50:55.473343 109.91… 178.33.… HTTP  635 POST /session_login.cgi HTTP/1.1  (application/x-
68854 2021-02-11 11:50:55.488388 178.33… 109.91.… TCP    86 808 → 53934 [PSH, ACK] Seq=3950 Ack=962 Win=16896
68874 2021-02-11 11:50:55.498728 178.33… 109.91.… HTTP  237 HTTP/1.0 302 Moved Temporarily
```

```
    Request Method: POST
    Request URI: /session_login.cgi
    Request Version: HTTP/1.1
  Host: [redacted]:808\r\n
  User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
  Accept-Language: de,en-US;q=0.7,en;q=0.3\r\n
  Accept-Encoding: gzip, deflate\r\n
  Referer: [redacted]net:808/\r\n
  Content-Type: application/x-www-form-urlencoded\r\n
> Content-Length: 33\r\n
  Origin: [redacted].net:808\r\n
  DNT: 1\r\n
  Connection: keep-alive\r\n
> Cookie: testing=1\r\n
  Upgrade-Insecure-Requests: 1\r\n
  \r\n
  [Response in frame: 68874]
  [Full request URI: [redacted].net:808/session_login.cgi]
  File Data: 33 bytes
v HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "page" = "/"
  > Form item: "user" = "root"
  > Form item: "pass" = "sikerim20"
```

Summary

Useraccount and password were found in the network traffic

We should talk about the „*security*“ of HTTP?

Summary

Useraccount and password were found in the network traffic

We should talk about the „*security*“ of HTTP?

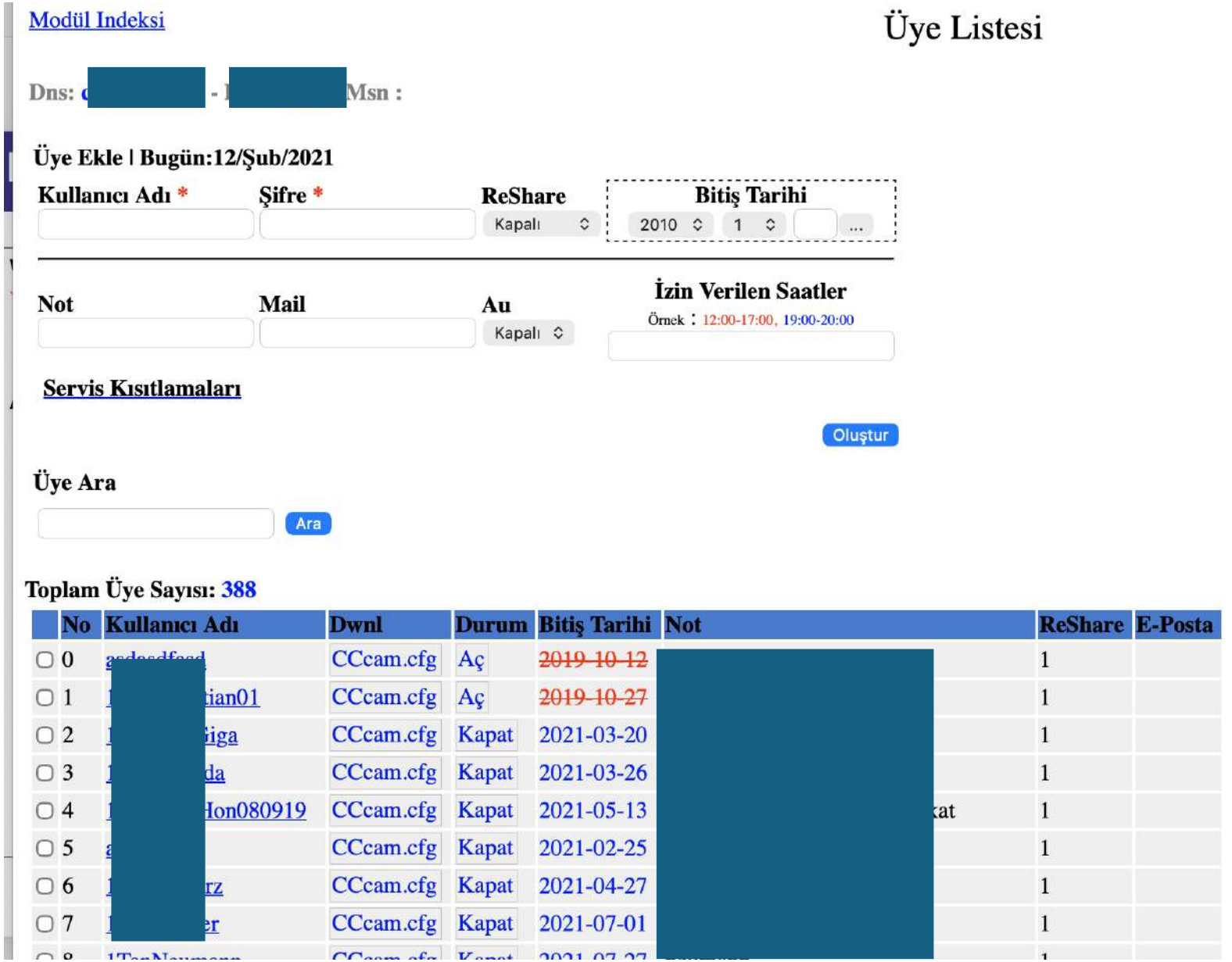
Same for server 2 (Kowboy, kowboy) and server 3 (usXXXX, SuSXXXX), but not for server 4

XXXX because username and password contain real name of the suspect ;-)

Did we discuss about passwort rules and security?

Final step - Finding customers

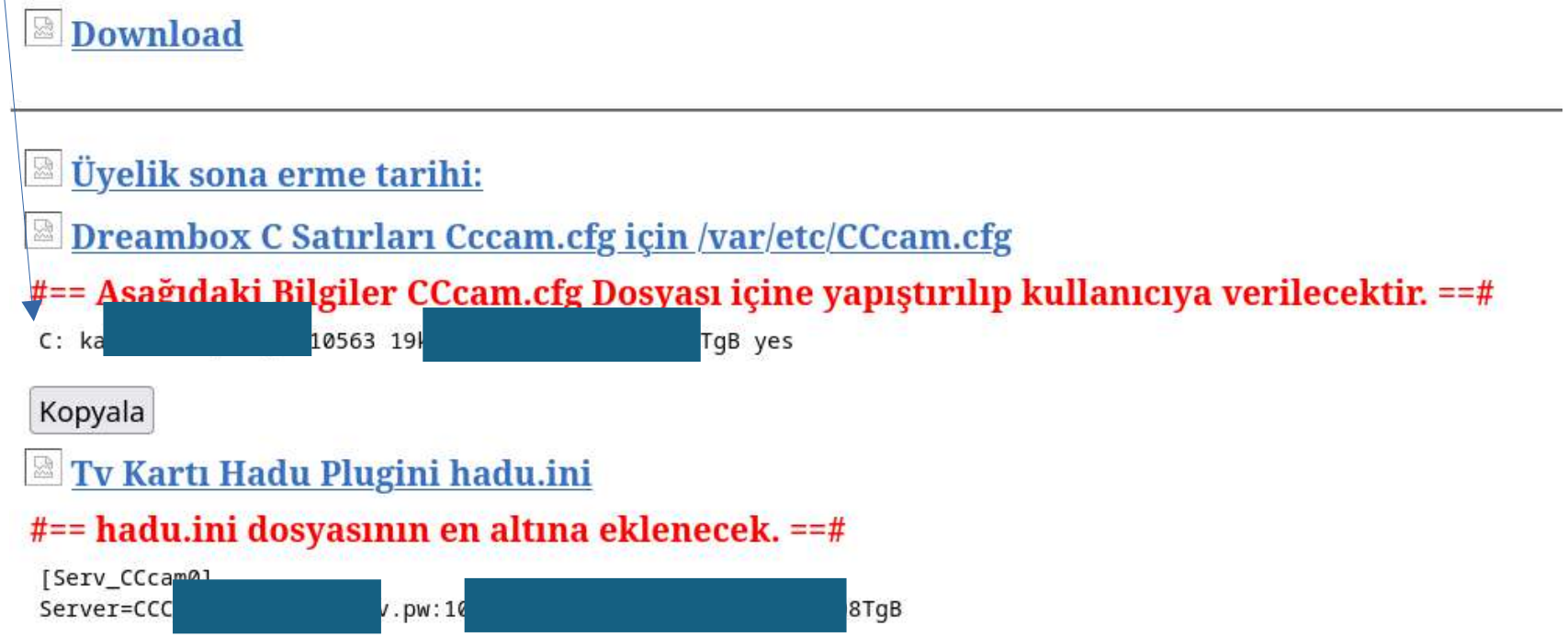
- Tcpflow extracts everything I needed
- Webmin was used for user administration
- Users can have different contracts with the suspects
- A contract determines how long the „service“ can be used



CCcam.cfg

- Each user has its own config-file

  (*This is a config file which is uploaded to the receiver)*
- Webpage allows download of a config file mapped to a user
- Config-file has a c-line which defines the server to get the decryption keys



Translation

Türkisch – erkannt | Deutsch | Englisch | Französisch

Üyelik sona erme tarihi:
Dreambox C Satırları Cccam.cfg için /var/etc/CCcam.cfg
#== Aşağıdaki Bilgiler CCcam.cfg Dosyası içine yapıştırılıp kullanıcıya verilecektir. ==#

C: kab63 19kabe608TgB yes

Tv Kartı Hadu Plugini hadu.ini
#== hadu.ini dosyasının en altına eklenecek. ==#

[Serv_CCcam0]
Server=CCCam:kab608TgB

Tv Kartı Acamd Plugini cardclient.conf
Octagon ve Next Chazlar İçin Bilgi

<- Geri Dön (Müşteri Listesi)

Deutsch | Englisch | Französisch

Membership expiration date:
Dreambox C Lines for Cccam.cfg /var/etc/CCcam.cfg
#== The following information will be pasted into the CCcam.cfg file and given to the user. ==#

C: kab63 19kabe608TgB yes

TV Card Hadu Plugin hadu.ini
#== Added to the bottom of the hadu.ini file. ==#

[Serv_CCcam0]
Server=CCCam:kab608TgB

TV Card Acamd Plugin cardclient.conf
Information for Octagon and Subsequent Devices

<- Return (Client List)

More details

- Tcpflow does not map everything correctly every time
- Deeper look into the payload unhides undeteced details:
  - Different packets were available
  - More PayTV-services were offered

```
<td><b>Paket</b><br><select
class='ui_select' name="package" >
<option value="" >
<option value="0d00:0:1, 0664:0:1,
092b:0:1" >Turkey Paket
<option value="093b:0:1, 0919:0:1"
>SkyIt Paket
<option value="0500:042200:1,
0500:041700:1, 0500:042700:1,
0500:043800:1" >Adult Paket
<option value="1702:0:1, 1722:0:1,
1833:0:1, 1834:0:1, 09c4:0:1,
098E:0:1, 09C7:0:1" >SkyDe Paket
<option value="0963:0:1" >SkyUk Paket
</select>
```

More a lucky guess than careful investigation

Results

- Detecting 4 different servers used for card sharing
- Detected system specific information to support the search
- Detected login information related to the suspect

- Found overall 3197 customer accounts, where 1750 were still active
- Determined 34 account modifications (adding, password changes, contract runtime) made during the LI

Summary

- Found server-names and ip-addresses

  These were later found in the firefox and chrome history.

- Found user agent

  Such a system was found during a later search, seized and investigated. The internet history showed connections to the server (see above).

- Found usernames and passwords

  Map to usernames and password used by the suspect.

  Found in password lists on the supects computer.

- Found content of Cccam.cfg files

  Same CCcam.cfg files were found on suspects system.

All of this made the accuse easier

Facts and figures

- Used tools:
  - Primary tools:
    - Wireshark / tshark 3.4.4
    - tcpflow 1.6.1
    - bash
  - Secondary tools (verification for second software principle)
    - Brim 0.24.0
    - Arkime 2.7.1
- Time for investigation:
  - *23.02.2021 – 20.03.2021*
  - *overall approx. 150h*
    - *Repeated analyses based on extracted information was time consuming*
    - *Better hardware would have improved this (I think so)*
    - *Some things might be solved smarter, faster, better with different tools or knowledge*

Facts and figures II

- Amount of data (moved, copied, merged, extracted, analyzed):
  - Nearly 4.5 TB (starting from approx 1.9 TB)
  - Approx 2.6 billion packets (starting from 1,593,482,440 captured packets)

- Relevant packets at the end:
  - 95,315 packets
    - Server 1: 15,321 packets
    - Server 2: 16,299 packets
    - Server 3: 63,672 packets
    - Server 4: 23 packets
  - 0.00598 % of captured packets

Conclusion

It is easy when you know what to look for...

SharkFest'25 EUROPE

Time for questions

Feedback:

Contact:
Daniel Spiekermann
Discord: lo_127001
daniel.spiekermann@fh-dortmund.de