



CODENOMICON

Fuzzing Embedded Devices

Finding unknown vulnerabilities
in home electronics

Rikke Kuipers and Ari Takanen



Hybrid III: designed to gather data
from frontal impacts

DEFEND. THEN DEPLOY.



CODENOMICON

Industry is Slowly Waking Up to the Unknown Threats

“All software has undetected exploitable vulnerabilities”
- Security Vendor 2009

“All our zero-day vulnerabilities were found with Fuzzing.”
– Software Vendor 2010

“You would be a fool not to Fuzz.”
– Analyst 2011

Segmentation fault (core dumped).





What is Fuzzing?

CODENOMICON

- A testing technique where purposefully unexpected and/or invalid input data is fed to tested system in hope to find robustness and security problems





Fuzzing Techniques

CODENOMICON

- Mutation/Template-Based Fuzzing
 - Quality of tests is based on the used template (seed) and mutation technique
 - Slow to execute, least bugs found
- Generational/Specification-Based Fuzzing
 - Full test coverage, as the model is built from specification
 - Fast to execute, most bugs found





Fuzzer Efficiency Case Study

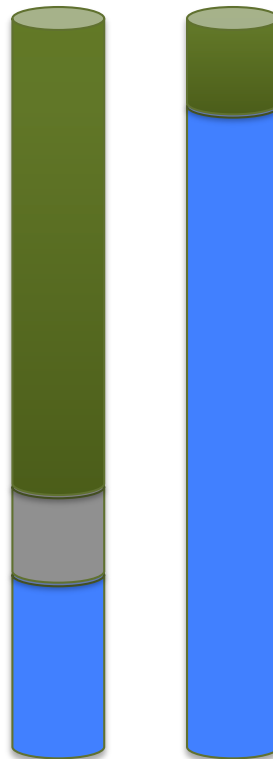
CODENOMICON

- Most important efficiency metric for fuzzers:
 - How many bugs does it find
 - How much time does it take to find them

“Smart” model-based
generational fuzzer
found 10 unique bugs

Both found 2 same bugs

Mutation fuzzer found
4 unique bugs



Generation fuzzer
executed for 17 hours

Mutation fuzzer took
118 hours (5 days) to
execute, **after which no
more new bugs were
found**



Codenomicon Labs

CODENOMICON

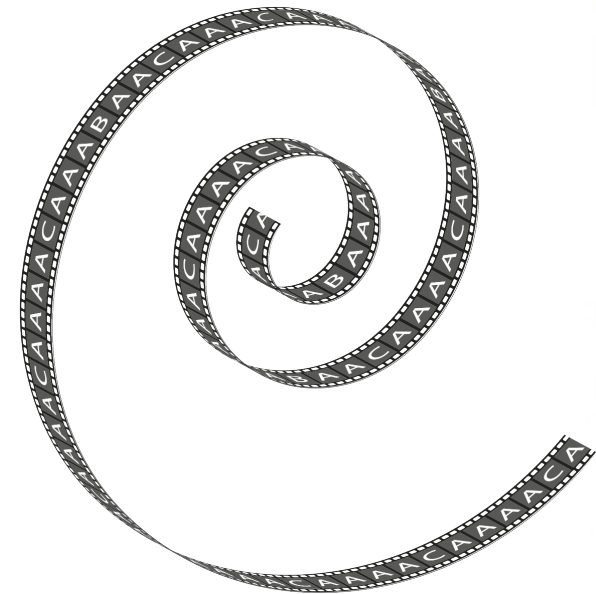
- We have tested and released test reports on:
 - WiFi access points
 - Bluetooth devices (including cars and medical devices)
 - NAS devices
 - Printers
 - Browsers
 - **Smart TVs**
- **Any idea what we should test next?**



TV Attack Vectors

CODENOMICON

- Dumb DVB-enabled TVs
 - DVB-C/T
 - IR
- Media center TVs
 - basic network connectivity: IPv4, UPnP, DLNA, DHCP, HTTP, FTP
 - Digital media: images, videos, audio
 - USB and memory cards
 - Bluetooth and WiFi (client)
 - limited network services
- Internet-enabled TVs
 - “Web 2.0 client”
 - Applets, applications, widgets
 - Full browser
 - **Capability very similar to smart phones**





CODENOMICON

Demo/Video about JPG fuzzing

The screenshot shows the Defensics 10 application interface. The sidebar on the left lists configuration steps: 1 Basic configuration, 2 Interoperability, 3 Advanced configuration, 4 Instrumentation, 5 Test cases, 6 Test run, 7 Results, and 8 Remediation. The main window displays test results for a JPEG file, showing a table of elements and anomalies.

| # | Element | Anomaly |
|----|---------------|----------|
| 0 | valid | none |
| 1 | image.element | built-in |
| 2 | image.element | built-in |
| 3 | image.element | built-in |
| 4 | image.element | built-in |
| 5 | image.element | built-in |
| 6 | image.element | built-in |
| 7 | image.element | built-in |
| 8 | image.element | built-in |
| 9 | image.element | built-in |
| 10 | image.element | built-in |
| 11 | image.element | built-in |
| 12 | image.element | built-in |
| 13 | image.element | built-in |
| 14 | image.element | built-in |
| 15 | image.element | built-in |
| 16 | image.element | built-in |
| 17 | image.element | built-in |
| 18 | image.element | built-in |
| 19 | image.element | built-in |
| 20 | image.element | built-in |
| 21 | image.element | built-in |

The right pane shows a detailed message log for the selected element:

```
#0
Index
Group
Hash
Messages

Message
000000 sequence
000000 image
000000 soi-marker
000000 SOI .. ff d8
000002 ExtraAnomalyAppSegment
000002 marker-segment
000002 Marker
000002 marker
000002 APPn .. ff #0
000004 Length .. 00 10
000006 marker-data
000006 app-segment
000006 jfif-segment
000006 JFIFIdentifier JFIF. 4a 46 49 46 00
00000b Version .. 01 01
00000d Units .. 01
00000e XDensity .. 01 2c
000010 YDensity .. 01 2c
000012 XThumbnail .. 00
000013 YThumbnail .. 00
000014 RGBThumbnail .. 00
000014 marker-segment
000014 Marker
000014 marker
000014 APPn .. ff #1
000014 Length .. * 18 2a
000018 marker-data
000018 app-segment
000018 exif-segment
000018 EXIFIdentifier Exif. 45 78 69 66 00
00001d Pad .. 00
00001e tiff-data
00001e ByteOrder MM 4d 4d
000020 TIFFIdentifier42 .. * 00 2a
000022 NextOffset .... 00 00 00 08
000026 tiff-td
000026 NumberOfDirectoryEntries .. 00 0e
```

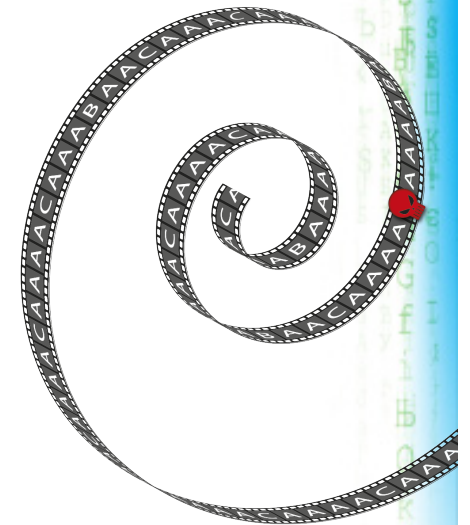




New attack vector: DVB

CODENOMICON

- Not just to transmit video/audio streams
 - vehicle to vehicle networks
 - navigation systems
 - handheld communications (DVB-H)
 - internet transport (IP-over-DVB/MPEG)
 - military (DVB-S2)
- DVB stream can contain several “channels” multiplexed into one stream, de-multiplexed at the receiver





Structure of MPEG2-TS / DVB

CODENOMICON

- Audio/video streams, in channel bundles
- Informational “tables” about the payload content, such as Program Association Table (PAT)

CAAAACAAABAAACAAACAAACAAAB

- Each type of frame or table needs to be fuzzed



DVB Fuzzing

CODENOMICON

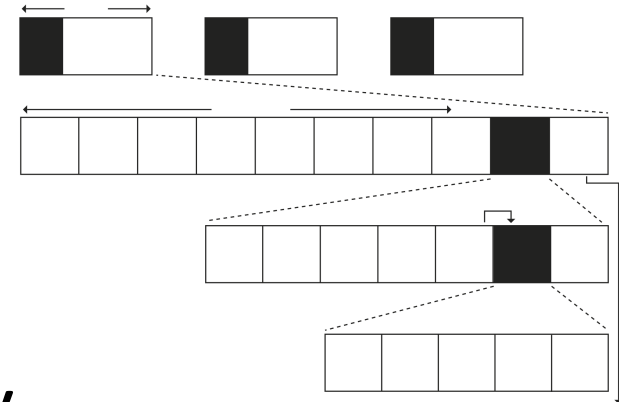
- Selection of tool:
 - Mutation fuzzing is the easiest, and most interoperable, but can be country-specific
 - Model-based, generation fuzzing, is more optimized to find bugs faster
 - Our solution was easy: Defensics MPEG2-TS fuzzer (available since 2010)
- Injection:
 - Our solution: after “file fuzzing”, multiplexing the stream back to right format, and then injecting using an off-the-shelf modulator



Fuzzing MPEG2-TS

CODENOMICON

- MPEG-TS acts like a protocol
- When fuzzing, each feature needs to be fuzzed separately



- “Common frames” require pauses between tests so that stream stays in sync

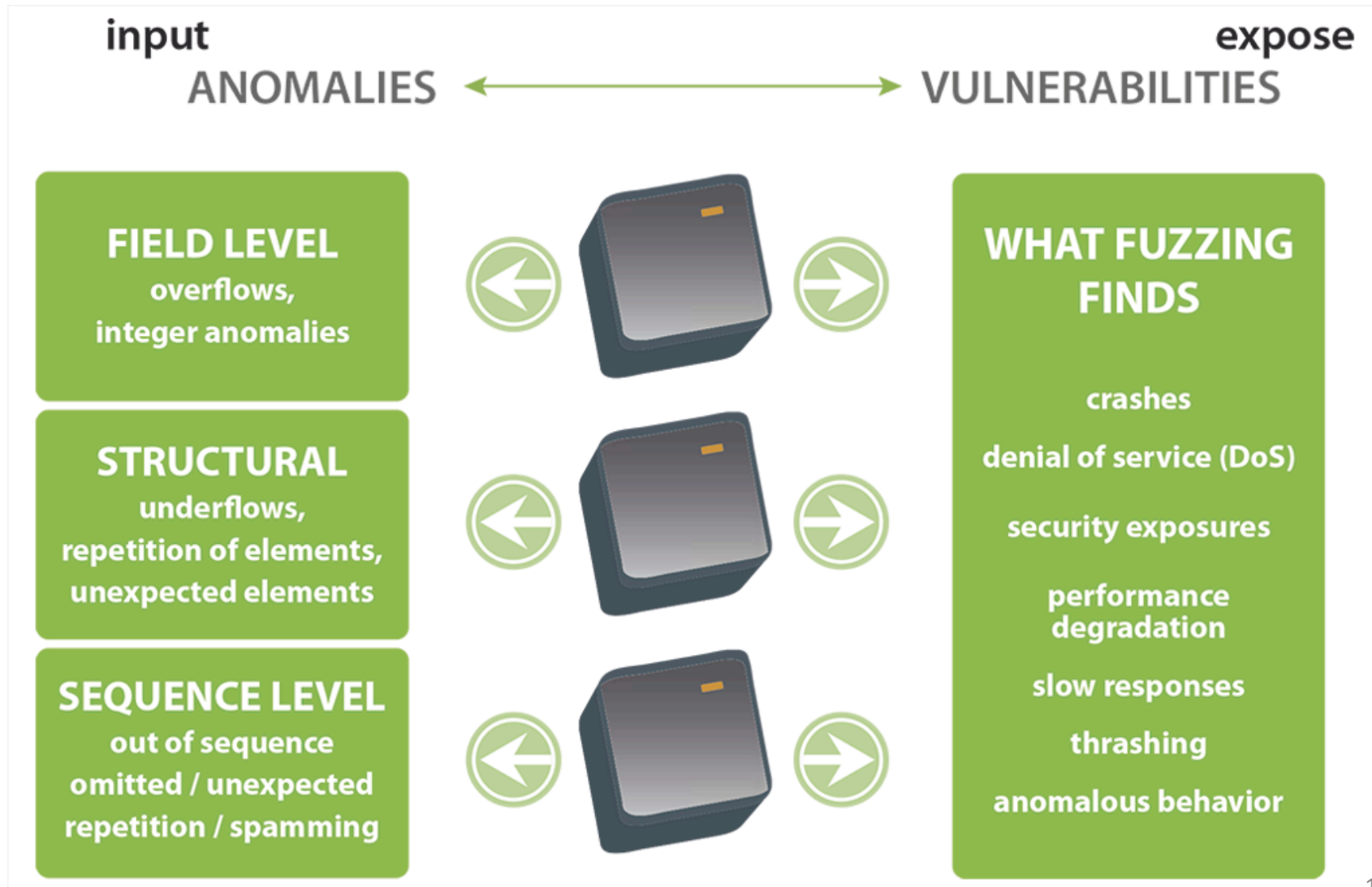


- “Rare frames” can require long streams so that all functionality is tested



Anomalization

CODENOMICON





Example Anomalies

CODENOMICON

The screenshot displays the CODENOMICON interface. On the left is a vertical navigation pane with eight steps: 1 Basic configuration, 2 Interoperability, 3 Advanced configuration, 4 Instrumentation, 5 Test cases (highlighted), 6 Test run, 7 Results, and 8 Remediation. The main area is titled 'Test cases' and shows a tree view under 'Full (200k cases)'. The tree structure is as follows:

- pid-0
 - psi-stream-pat
 - psi-section-pat
 - element
 - psi-section
 - element
 - table-id
 - section-syntax-indicator
 - private-indicator
 - reserved
 - section-length
 - section-data
 - crc32

Below the tree is a dropdown menu showing 'MPEG.transport.ts-packet-payload.ts-packet'. At the bottom, a table lists anomalies:

| # | Element | Anomaly |
|---|---|----------|
| 1 | transport.ts-packet-payload.ts-packet.element | built-in |
| 2 | transport.ts-packet-payload.ts-packet.element | built-in |
| 3 | transport.ts-packet-payload.ts-packet.element | built-in |
| 4 | transport.ts-packet-payload.ts-packet.element | built-in |
| 5 | transport.ts-packet-payload.ts-packet.element | built-in |



Example Anomalies

CODENOMICON

1 Basic configuration Test cases

| ➔ Message | | | |
|-----------|---------------------|----------------------|-------------|
| 000000 | stream | | |
| 000000 | ts-packet-payload | | |
| 000000 | ts-packet | | |
| 000000 | ts-prefix | | |
| 000000 | sync-byte | | G 47 |
| 000001 | trans-error | | 1bit 0 |
| | pay-start | | 1bit 0 |
| | trans-prio | | 1bit 0 |
| | pid | 13bit 00000 00110001 | |
| 000003 | tsc | | |
| 000003 | not-scrambled | | 2bit 00 |
| | afc | | |
| | adaptation-payload | | 2bit 11 |
| | ctr | | 4bit 1111 |
| 000004 | adaptation-payload | | |
| 000004 | ts-adaptation-field | | |
| 000004 | length | | * <u>2a</u> |
| 000005 | ts-payload | | |
| 000005 | ts-payload-data | | () |
| 000005 | stuffing | | () |

➔ Message

CODE

| | | | |
|--------|--------------------------|------------------------|------------------------|
| 000000 | stream | | |
| 000000 | pes-stream-ac3 | | |
| 000000 | pes-packet-ac3 | | |
| 000000 | ... | 00 00 01 | |
| 000003 | stream-id | | |
| 000003 | private-1 | . bd | |
| 000004 | length | .. <u>00 01</u> | |
| 000006 | pes-packet-header | | |
| 000006 | | 2bit 10 | |
| | scrambling-control | | |
| | not-scrambled | 2bit 00 | |
| | priority | 1bit 0 | |
| | data-alignment-indicator | 1bit 1 | |
| | copyright | 1bit 1 | |
| | original-or-copy | 1bit 1 | |
| 000000 | 000007 | flags | |
| 000000 | 000007 | pts-dts | 2bit 10 |
| 000000 | | escr | 1bit 0 |
| 000000 | | es-rate | 1bit 0 |
| | | dsm | 1bit 0 |
| | | aci | 1bit 0 |
| | | crc | 1bit 0 |
| | | ext | 1bit 0 |
| 000000 | 000008 | length | . 05 |
| 000000 | 000009 | variable-length | |
| | 000009 | | 4bit 0010 |
| | | presentation-timestamp | |
| | | base-32-30 | 3bit 101 |
| | | mbit | 1bit 1 |
| 000000 | 00000a | base-29-15 | 15bit 00100101 0011100 |
| 000000 | | mbit | 1bit 1 |
| 000000 | 00000c | base-14-0 | 15bit 11001100 1010110 |
| 000000 | | mbit | 1bit 1 |
| 000000 | 00000e | stuffing | () |
| 000000 | 00000e | ac3-bitstream | () |
| 000000 | 00000e | ac3-synctrain | () |
| 00000e | | ac3-svncinfo | |

G 47

1bit 0

1bit 0

1bit 0

0001

bit 00

bit 11

1111

* 2a

()

()



Modulation

“Thanks Sofia
Digital!”

CODENOMICON

```
p0c@code:~$ DtPlay stream.ts -n 1 -t 215 -ml -27.5 -i 1 -mt OFDM -mC QAM64 -mf  
530 -mG 1/8 -mc 2/3 -mT 8k -r 22100000
```

- n Device number to use: **1**
- t Device type to use: **215** (DTU-215)
- i Port number of the output channel to use: **1**
- r Transport-Stream Rate in bps or sample rate in case of IQ-modulation mode:
22100000
- ml Output level in dBm: **-27.5**
- mG DVB-H/DVB-T guard interval: **1/8**
- mc Convolutional rate: **2/3**
- mf Modulation carrier frequency in MHz: **530**
- mt Modulation type: **OFDM**
- mC ATSC/DVB-H/DVB-T/DTMB constellation: **QAM64**
- mT DVB-H/DVB-T transmission mode: **8k**





CODENOMICON

Interoperability and Instrumentation

- Before fuzzing, the features in the target device need to be scanned
- Valid sequences are the easiest method, by building valid traffic directly from protocol specification
- Target device can be “picky” on what data it will accept
- Our solution: Capture of the local national TV stream is fed to the fuzzer

- In DVB, you cannot use valid sequences for instrumentation, as tests are unidirectional
- ICMP heart-beat “ping” is a simple instrumentation



Results

CODENOMICON

| Protocol/ TV | TV 1 | TV 2 | TV 3 | TV 4 | TV 5 | TV 6 |
|-----------------|-------------|-------------|-------------|-------------|-------------|-------------|
| IPv4 | pass | FAIL | FAIL | pass | pass | FAIL |
| DVB | FAIL | FAIL | FAIL | FAIL | FAIL | FAIL |
| UPnP | n/a | FAIL | pass | n/a | n/a | FAIL |
| Images | pass | FAIL | FAIL | n/a | n/a | FAIL |
| Audio | pass | pass | n/a | n/a | n/a | pass |
| Video | FAIL | FAIL | n/a | FAIL | FAIL | FAIL |

“**FAIL**” means multiple repeatable crashes were found

“pass” means the system did not crash (more fuzzing needed?)

“n/a” means the interface did not exist, or was not tested

We did not analyze the failures for exploitability.



Analysis

CODENOMICON

- As far as we know, there are no other fuzz tests against TVs
- No IPv6 yet in any TV
- Bad quality IPv4 still around
- DVB was easiest attack vector, probably because no DVB fuzzing available before this
- Video testing was with one container/codec only, but still lots of failures
- **TV fuzzing might not be high priority, but most likely same DVB/MPEG codecs are also used in other industry domains**



CODENOMICON

THANK YOU!

QUESTIONS to:

ari.takanen@codenomicon.com

“Thrill to the excitement of the chase!
Stalk bugs with care, methodology,
and reason. Build traps for them.

....

Testers!

Break that software (as you must) and
drive it to the ultimate
- but don't enjoy the programmer's
pain.”

[from Boris Beizer]