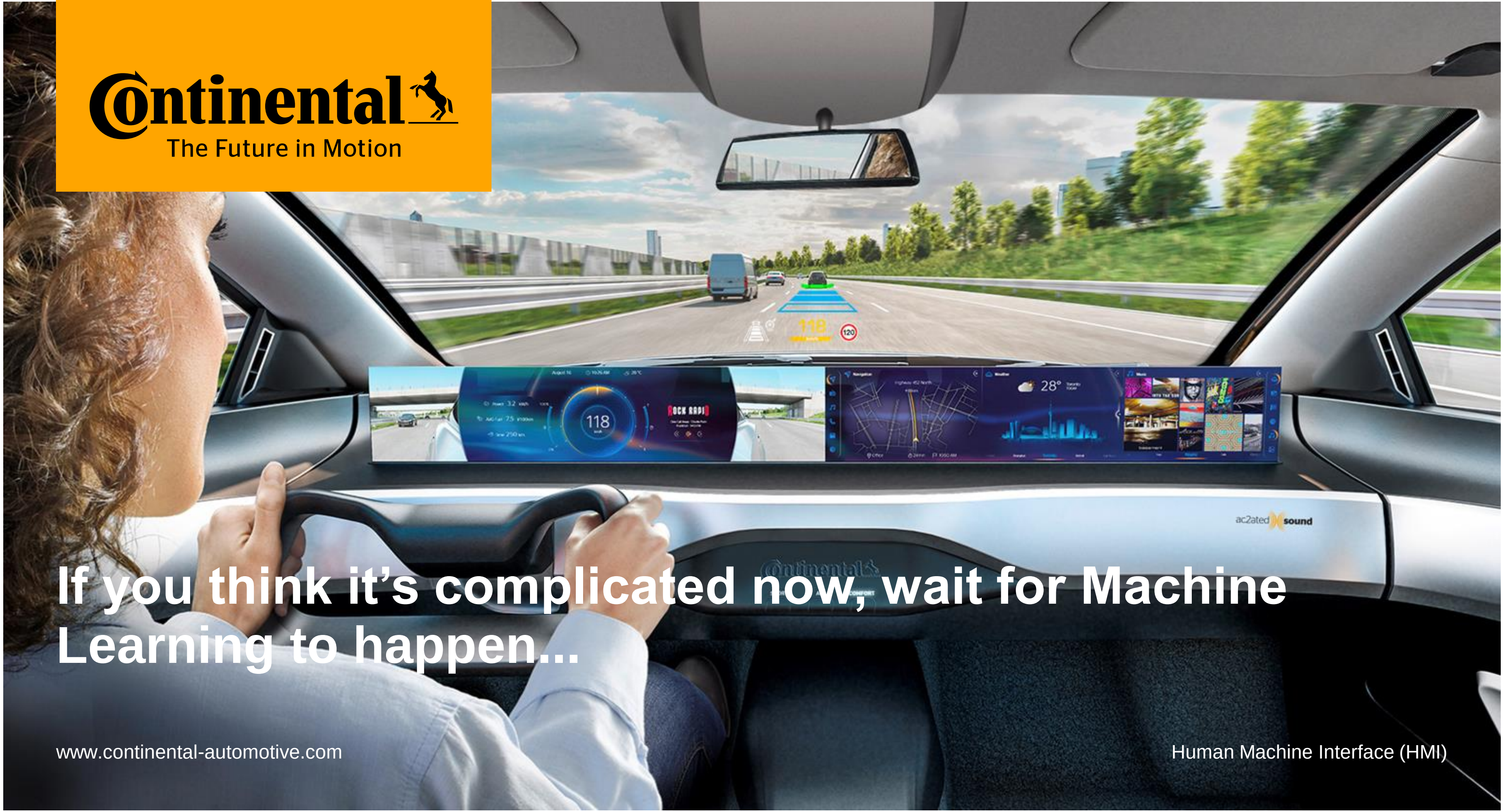
Continental

The Future in Motion



# If you think it’s complicated now, wait for Machine Learning to happen...

www.continental-automotive.com

Human Machine Interface (HMI)

# ABOUT ME

# Victor Marginean

Born in Romania – grew up in a city with one citadel & Hollywood like sign.
Worked for 12 years in Telecommunication, Cloud Computing and AI then moved to Automotive.
Radio Network Engineer and several customer facing roles for Chinese, French and US customers.
Moved to Germany in 2018 @ Continental BU HMI in Babenhausen.

# WE DO HMI

Display Solutions

Head Up Displays

Interior Cameras

Instrument Clusters

Full Digital Clusters

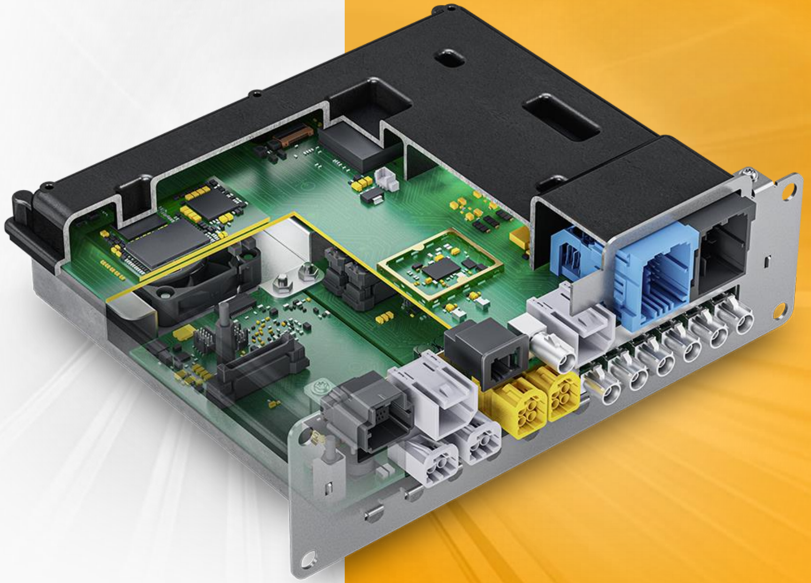
Cockpit HPC

Sound Solutions

WE **DO** HMI

### Secure Products

# Start with Boot ROM



Boot ROM is a small piece of mask ROM or write-protected flash embedded inside the processor chip. It contains the very first code which is executed by the processor on power-on or reset.

Vulnerabilities inside the BootROM can’t be fixed with SW Updates, therefore it needs to be thoroughly checked via:

- Secure code reviews
- Fuzz testing the Boot Rom SW

The additional complexity is added by the fact that usually suppliers are not providing their source code.

**Secure Products**

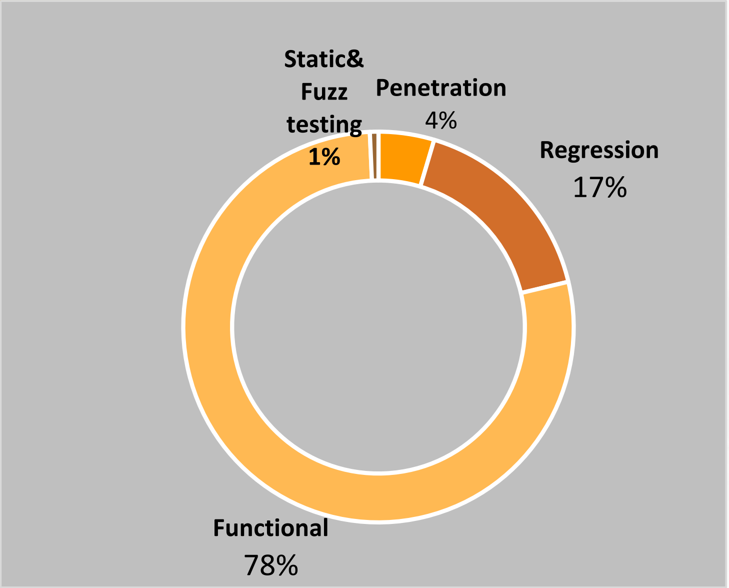
# Continue with everything else

WE **DO** HMI

Several layers are needed for ensuring a working & secure Product. For testing, the following is needed:

- Static Testing
- Functional Testing
- Regression Testing
- Fuzz testing
- Penetration testing

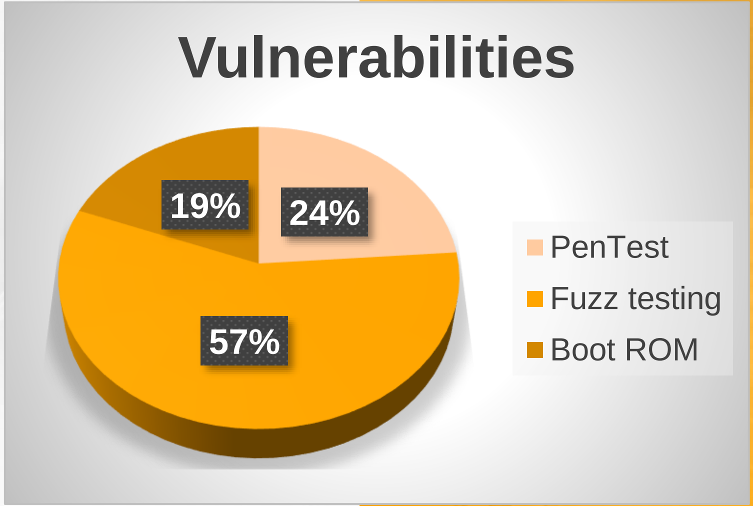
Best way to make sure the product is properly tested is to give directly to the developers the tools for checking & fixing their code at an early development stage.



WE **DO** HMI

### Secure Products

# What did we get from the 1%



Vulnerabilities were discovered and fixed in all cases: Boot ROM Secure Code Review, Fuzz testing and PenTest.

Vulnerabilities discovered by the 3 methodologies were different because of the SW code available for testing.

The IP protection introduces additional hurdles for security testing end to end.

## Secure Products

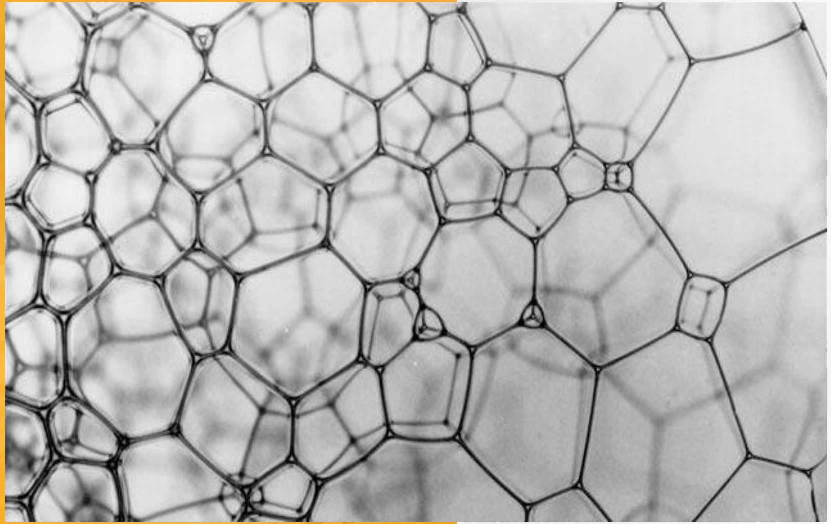
# SW Complexity

WE **DO** HMI

The SW is an ecosystem of hundreds to thousands of functions „living“ inside a micro-controller influencing the mean time between failures (MTBF).

In order to understand a potential evolutionary trajectory we should have the means to measure the interactions between different functions to see which combinations are most fit.

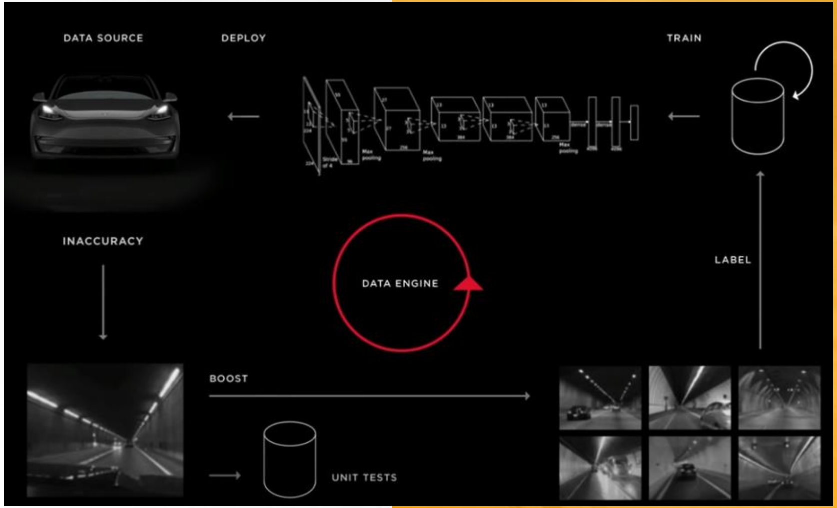
A software that is evolving from features point of view should also take the most secure path.



WE **DO** HMI

## Secure Products

# Why worry?



This is an example from a public presentation coming from a famous car manufacturer.

There is no mention about some sort of Security testing.

The industry seems to be focused now on making the products work.

And this is now NORMAL.

But what will happen when ML will be controlled by the attacker?

Secure Products

# How do we prepare for ML?

WE **DO** HMI

The additional attacks that come from machine learning arrival to Automotive are adding to the complex SW that needs to run under already stringent Automotive requirements.

Testing methods are needed for covering the different attacks:

- Adversarial Examples
- Data Poisoning
- Online System Manipulation
- Transfer Learning Attack
- Data Confidentiality

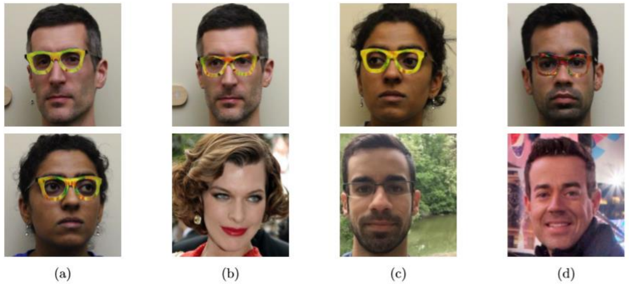


Figure 4: Examples of successful impersonation and dodging attacks. Fig. (a) shows $S_A$ (top) and $S_B$ (bottom) dodging against $DNN_B$. Fig. (b)–(d) show impersonations. Impersonators carrying out the attack are shown in the top row and corresponding impersonation targets in the bottom row. Fig. (b) shows $S_A$ impersonating Milla Jovovich (by Georges Biard / CC BY-SA / cropped from https://goo.gl/GlsWlC); (c) $S_B$ impersonating $S_C$; and (d) $S_C$ impersonating Carson Daly (by Anthony Quintano / CC BY / cropped from https://goo.gl/VfnDct).

WE **DO** HMI

### Secure Products

# Even more Smart Fuzzing?

Considering the increasing complexity of Automotive SW, Smart fuzzing might be our chance to discover and fix the mutations that could be induced by an attacker.

This might be achieved if an evolved Smart Fuzzing is:

- Integrated in the CI system for the adversarial examples (e.g. as part of the testset)
- Run for each Pull Request
- Developers/Neural network fixes the vulnerabilities found

WE **ARE** HMI

# DRIVING THE TREND.