1. What is the basic mechanism of defeat software? How could it detect the difference between type-approval testing and actual driving? According to your knowledge, how many different types of defeat systems are in use (thermal window, open hood, detection of the cycle...)?

I'm not aware of a defeat device in the current discussion that uses a condition that's inherently different in a lab versus on the street, such as open hood or physical acceleration. (An exception to this is the steering wheel detection that Volkswagen added at a late stage in the US.)

Most defeat systems that I've looked at are either detecting the cycle directly (as in the case of Volkswagen, where a distance-over-time relationship was verified with the reference cycle data), or indirectly (by verifying a number of parameters that are very likely to only be satisfied completely during a test cycle, such as an engine load threshold, temperature windows, atmospheric pressure).

Yet other systems are purely time-based, i.e. change the behavior of the car after a time that corresponds with the length of the test cycle.

2. How did you do the hacking or reverse engineering, technically speaking? How much effort does it take, how much time, which hardware do you need? How did you detect the on board diagnostics data streams? How do you know which data belongs to which parameter? Was the software encrypted? If we understand correctly, you basically found software that could detect the speed in the NECD test. As soon as these speed parameters were detected by the ECU software the emissions technology of the vehicle were reduced in their operation. Did you also find other ways for cycle beating? During the hearing of Volkswagen, the company claimed that the car has to detect being placed in a (lab) test cycle in order to disable certain functions unrelated to emissions, such as collision control. What information is required to prove the test cycle detection is not used to defeat the emissions test?

The steps of reverse-engineering the defeat device in the Volkswagen engine ECU was very similar to any other reverse-engineering task:

The first step is to obtain the software image. I did this with a combination of exploiting a hardware vulnerability in the Infineon TriCore chipset and knowledge obtained observing in-official and official service tools. Other methods (which I didn't use) include intrusive techniques, for example a decapsulation of the chip containing the firmware, or obtaining the software from in-official flash tools such as those used by aftermarket tuning industry. In either way, the result is an un-encrypted, but still only computer-readable software image that can be turned into a human-readable text with a so-called "disassembler". This does not yield the original source code that was used to develop the device, but an intermediate step in so-called "assembly" language. Any identifiers, such as variable or function names, comments as well as some structure is lost during the original process of "compilation", and can't be recovered.

The second step is to understand the software. Because of the computationally-driven nature of the ECU (no identifying strings, no single-step debugging feasible etc.), I extensively produced log data (such as memory snapshots) while driving regularly. In my case, this was possible over the OBD-II interface using vendor-specific commands without having to open or modify the ECU.

The following step involves identifying variables, starting with basic ones such as rpm, speed, temperatures, and eventually emission-related data. The goal is then to understand more about the algorithms that control the emission management. Usually this involves identifying functionality related to emissions control (such as SCR) first, which is usually at least partially encapsulated in dedicated functions.

At this point, connections between modules can be identified. For example, the "acoustic function", which detects the driving profile, is encapsulated in the "injection strategy" block that computes injection amounts based on torque and other parameters, which are also adjusted when a test cycle is detected. It can be seen that the same variable (that indicates if a test cycle is currently active) is used in a function related to SCR functionality.

Detail analysis of the affected functions show how the SCR functionality behaves fundamentally different in a test cycle.

No connection from environmental parameters, such as the indication if rear wheels are moving, or physical acceleration is present etc. that feeds into the SCR or EGR functionality was found, which shows that these criteria are not used to modulate the emissions efficiency.

Mistakes in the manual process of reverse-engineering such as to misinterpret functionality when just looking at individual functions must be avoided. Every result must be validated by observing real driving data, and verifying that the car behaves in the predicted way.

3. Based on the structure of the ECU explored in your VW Sharan, is it possible to differentiate between the standard software developed by Bosch and the specific requirements customized exclusively for VW? If so, how flexible was the standard software with regard to the later adaptation by the car manufacturer? Can you imagine that individual suppliers of standard software compete by offering more flexible software than their business rivals?

In the case of the Bosch EDC17 (the ECU series that is used in many of the affected Volkswagen cars), Bosch provides a well-documented software that is mostly data-driven. The data, sometimes called "calibration data", can affect large parts of how the software works; improper calibration data can cause engine damage, out-of-spec emission behavior, whereas fine-tuning this data allows to significantly improve emissions and performance. As such, when evaluating the functionality of the ECU, the software and data parts must be considered to be one piece.

According to a few internal documents that were ultimately published, Volkswagen supplies some parts of the software; examples being the "Start/Stop" functionality, the tachometer display optimization etc. This can be seen from the software as well, because these Volkswagen parts are located in a specific area in the binary code (at the end), by the nature of how the code is linked together in the software development.

Based on my information, Volkswagen does not have access to the complete sourcecode of the Bosch implementation. Instead, they obtain sufficient documentation that allows them to tweak the calibration data to their needs. Explicitly, the emission control code was not delivered by Volkswagen, but was implemented by Bosch.

A lot of "dead-coded" functionality is present in the code, i.e. algorithms that are computed, but the data part is setup so that the results are ignored. This is a standard procedure to allow the car manufacturer to select from individual algorithms without having to request software changes, or to unify software between different setups. As an example, SCR functionality is present in some ECUs that are used in cars without SCR hardware, but all SCR functionality is logically disabled in the data part. However, algorithms that are not in the code part cannot be added by the data part.

In the case of the "acoustic function", i.e. the function that detects a driving profile, the understanding is that a function was implemented by Bosch that detects an arbitrary cycle based on a time/distance relationship encoded in a curve in the data part. However, no data that would match a real test cycle data has been programmed into these curves by Bosch, just dummy data. Volkswagen then changed these parameters (along with thousands of other parameters for other functionality), and inserted the values to detect the NEDC.

Similarly, Bosch prepared a code path from the "acoustic function" to the emission control system, but did not activate the path in the data part; however, they provided Volkswagen with the necessary tools and documentation to enable this path.

It is my strong understanding that all functionality, included functionality that was only enabled with the data changes, was at any point in time well understood by both the supplier as well as Volkswagen.

4. You claim to have found defeat devices in Opel Zafira and Astra models. Opel claims that you have only selected four parameters out of 17 000 which you interpreted wrongly without taking into account physical parameters. Which criteria did you use as a basis for your investigations? Has your team used the Regulation (EC) No 715/2007 to interpret the situation? Did you have contact with representatives from Opel to discuss the case and did you receive the written reaction by the manufacturer in order to compare it with your findings?

The Opel Zafira 1.6 CDTi (EU6) was found to have a particularly high gap (roughly upper third from the tested cars) between the emissions found in the NEDC and PEMS measurements in NEDC variations (different temperature, +10% etc.) as well as other, more realistic driving scenarios. These gaps were found both in our own measurements as well as the German "Volkswagen investigation commission" report (http://www.bmvi.de/SharedDocs/EN/Pressemitteilung/2016/051-dobrindt-volkswagen-investigation-commission-report.html), as well as in a recently released DUH report. Also, measurements have shown a number of "hard" switches (instead of a ramped behavior) at certain temperature and speed limits that affect emission control; also some initially unknown variables were found to modulate the efficiency of the emission control scheme.

My work was to explain these observations by correlating the taken measurements to the algorithms found in software; for example, we knew that there was a dependency on temperature, so my work was to derive the exact temperature limits, as well as the impact to engine operation and emission control if these limits were exceeded.

Further, I've looked at which other conditions (that we didn't control initially in the dynamometer testing) would cause similar changes in operation, and worked out a driving profile that would demonstrate such behavior, and predicted what the change in operation would be based on the software.

Testing on a dynamometer then allowed us to verify that these condition indeed exist as predicted and cause a change in the emission algorithm; in this particular car, mostly a reduction in SCR efficiency (reduced or disabled AdBlue injection for a long time, causing tailpipe NOx emissions to increase by up to an order of magnitude) or an increase in raw NOx-emissions (due to smaller EGR rates than these found in the test cycle).

One criterion, from which a subset was previously explained as "rpm limitation", contained an algorithm that watches the engine load (derived from torque), and compares it with a threshold that was computed from a predetermined curve based on the current engine speed; multiple such curves were included in the software and selected based on the currently selected gear. This was a very interesting pattern to see, as from the perspective of engine protection the currently selected gear should not have any effect on engine behavior at a given rpm/torque operation point. Also, an excessively conservative hysteresis was implemented that prevented switching back when arriving at previous operating conditions.

Experimental testing showed that these programmed load thresholds closely corresponded (within a +10% tolerance) to the observed loads in the NEDC. Exceeding these limits even just temporarily causes the engine to emit higher emissions (because of a lowered EGR rate) from this point on, even when returning to the same load levels as before for a long time. The only way to reset to the EU6-compliant behavior is to go back to idle (lower than 1200 rpm) once. Any further acceleration over the programmed limits though will again switch to the non-emission-optimized behavior. Real-world driving tests showed that even with conservative driving the engine almost always operated in the non-emission-optimized behavior due to this load detection.

Other criteria includes temperature windows centered around the test cycle requirements (for example: programmed temperature range 17...33°C, required temperatures during testing is 20..30°C), a hard 150km/h speed limit at which AdBlue injection is switched off, as well as barometric pressure limitations that would reduce AdBlue injection above 850m over sea level.

All these criteria (except for barometric pressure) were validated on a dynamometer to exist and each yielded measureable and very often significant (i.e. greatly exceeding EU6) changes in tailpipe NOx emissions. We do not claim to understand the full extent of all parameters of the ECU operation, but we understand the subset of the criteria that modulates emission control and and verified their impact to tailpipe NOx emissions.

Our team found these algorithms to sense temperature, vehicle speed, engine speed (RPM), transmission gear and other parameters for the purpose of modulating, delaying or deactivating parts of the emission control system, with the result of a reduced effectiveness of the emission control system under conditions which may reasonably be expected to be encountered in normal vehicle operation. Per (EC) No 715/2007, these algorithms hence implement a defeat device.

This alone of course isn't sufficient to claim that an *illegal* defeat device exists, as these reductions could be necessary to satisfy engine protection requirements which were unknown to us. We did reach out to the manufacturer and asked them to explain these criteria, and why a reduction in efficiency is necessary in the described operating conditions. As an example, we asked why the SCR efficiency was lowered at a specific barometric pressure, independent from the (physically well-understood) barometric pressure threshold for EGR. Opel did unfortunately not provide any detail explanations, other than the remark that their cars do not contain any "illegal defeat devices" (see "Statement by Opel Group CEO Dr. Karl-Thomas Neumann on the Current Diesel Discussion", Opel Pressroom Europe).

Opel explained later, in a response to “DER SPIEGEL”, that the software does not implement a "test cycle detection" because the cars behave identically in the lab and on the street, provided that they are driven under the same conditions that exist during the NEDC testing (“DER SPIEGEL”, 23/1026, Page 70). This is an interesting statement, because this is true for the well-known Volkswagen defeat device also - if the NEDC is driven on the street, under similar environmental conditions like temperature and barometric pressure, a Volkswagen car would indeed activate the same "test mode" with full emission control efficiency that would be activated in a lab NEDC test.

In summary, I want to stress that, to the best of my knowledge, all my observations that have been published in this case are correct, are not intentionally misleading, and that the published interpretations and conclusions of our team based on these observations are correct. I cannot personally comment on whether the found algorithms conclude an *illegal* defeat device or not by law.

A recent report from the DUH ("NOx - und CO2 -Messungen an Euro 6 Pkw im realen Fahrbetrieb", http://www.duh.de/, 2016/09) compares PEMS measurement between a number of cars. The Opel Zafira 1.6 CDTi (2015), containing the analyzed software (with the defeat devices as described above), yielded an average NOx emission of 404 mg/km (5.1x EU6 limit), in-line with the measurements from the BMVI report.

Opel provided the DUH with another Zafira 1.6 CTDi (2016), which contained a modified software (but with the same engine) with optimized emission performance; the intention is to installed this software in a voluntary recall in existing Zafira 1.6 CDTi cars. The optimized software provided significantly better emissions at only 94 mg/km (1.2x EU6 limit). This shows that there is a significant potential in software improvements that affect real-world emissions. I would expect that these optimizations include the removal of at least some of the found defeat devices though we did not yet have a chance to analyze the updated software.

5. To your knowledge, did any authority commence criminal investigations following your revelations about the Opel Zafira and Opel Astra? If not, why do you think they have not done it so far?

I do not want to comment on this question.

6. You have found a file dealing with abatement system control under a pack of audio files which raises suspicions on the real intention of the carmaker. Don’t you think that carmakers can always hide source files in the reserved inaccessible partition under the pretext to defend industrial reserved info?

The name "acoustic function" was clearly a code name; while the original intention was indeed related to acoustics (even though it was the function that disabled the acoustic improvements in certain conditions to save fuel), the name alone doesn't properly explain what the function was doing: detecting a specific driving profile.

The word "acoustic function" is normally not visible outside of the car manufacturer. Only due to released internal documents the name that was found to be used in development. When doing a binary-only analysis using reverse-engineering (i.e. strictly only using information that is present in a given ECU, without any external information), no names a present, so any function is just defined by the actual computation. In that sense, for a binary-only analysis, it doesn't matter how the function is called; instead only the actual behavior is relevant.

However, usually (especially official reviews, for example during the certification process) analysis doesn't happen on binary level, but on sourcecode (or even higher) level, heavily relying on correct documentation and descriptive function names. Code names such as "acoustic function" could mislead reviewers. Any review should take into account that the documentation or internal naming may be deceptive, and should rely more on analyzing functionality than analyzing the naming. This, however, makes reviewing ECU software an enormous task as not only the intended behavior (as described in the software) must be analyzed, but also actual behavior as implemented in the binary image that is loaded onto the ECU.

7. Are you familiar with the chip tuning (modifying an erasable programmable read only memory chip in vehicle's electronic control unit (ECU) to achieve superior performance, be it more power, cleaner emissions, or better fuel efficiency)? What do you think about this type of modification to the ECU?

In a way, "chip tuning" is the logical extension to general car tuning which has existed since enthusiasts worked on cars. There is a significant safety risk involved in any kind of modification of critical powertrain equipment; this includes hardware as well as software. Because of this, there are regulatory hurdles in order to re-certify a tuned car; taking aside the illegal practice of not registering critical modifications, there is an existing process in place of how modified cars can be re-certified and legally used on streets.

Cars are optimized for certain targets which may not necessarily align with the customer targets; examples being longevity versus performance, or drivability versus absolute fuel efficiency. It is understandable that car manufacturers are interested in denying warranty or goodwill when a car was tuned, as components could be over-stressed and fail early. But some car manufacturers also offer software-based tuning packs. This yields to a financial incentive to shut down access to the ECU software using integrated security measures to lock out third-party tuners from their market.

As a side effect of these added protections, it's harder to do analysis on the software, so it's easier for car manufacturers to add "hidden" functionality.

Technically, it is feasible to allow "read-only" access to the software, but still prevent modifications. To a certain extent, this is currently implement in most vehicles (which allow read access to internal memory, but no write access to either volatile memory or non-volatile memory). If such a read access is restricted further, validating that the software works as expected is significantly harder.

With enough financial effort though, extracting and modifying the software will always be possible; ECUs are optimized for high-availability and low-cost, and as such can't provide very good tamper-resistance. Any currently deployed protection schemes will not prevent a commercially interested, financially backed party to reverse engineer the software, for example to illegally copy parts of the software. Improved protection schemes, however, make it more difficult, and under certain circumstances illegal, to provide an independent software analysis based on reverse-engineering by non-well-funded individuals.

8. In your opinion, how should emission tests be designed to avoid cheating? Do you think the RDE-tests are robust enough? Do you think it is possible to cheat testing with PEMS?

I expect that RDE test results correlate much better with the actual environmental impact of a car. But in my opinion, a test needs the manufacturer to present objective evidence that the emission strategies used in the certification test are in principle the same as those which are used in the field. Long-term statistics (such as which emission strategy has been used for how long) could be collected in regular intervals, and compared with the reference statistics collected in the certification process.

This requires the software to be honest in reporting the statistics, so this requires an effort of transparency, for example with expert reviews of relevant software code and engine behavior. But it no longer requires analyzing the exact algorithms for controlling the engine behavior (and for example determining if a defeat device is legal in order to protect the engine); as long as the engine, on average, operates similarly to the RDE, and produced good test results in the RDE, it would also not exceed emissions in the field.

9. In your answer to written question 9 for the hearing of Mr Daniel Lange, you mention that even with an RDE test there remains an incentive to circumvent the test. Additionally, there would be detection criteria, such as additional backpressure on the vehicle (related to the weight of the PEMS device). Do you have evidence that this signal for back weight could be useable? Are these data measured in vehicles? Why?

I'm not aware that cheating in RDE and/or PEMS measurements has been implemented, so this remains - hopefully - a purely theoretical scenario for now. However if this scenario isn't taken into account when designing tests, the incentive to build such cheats is increased as there would be no method to detect them. Ultimately the goal must be to verify that the long-term operation of the car matches the behavior during the certification test.

10. How feasible is it for car manufacturers to update ECU software over the air to run defeat device functions based on certain conditions, such as proximity to test labs or time frames in which testing is being suspected while at the same time reducing the risk of detection via reverse engineering of software permanently available on the ECU in individual cars? What is the most effective policy or technology to prevent this approach?

Technically it is possible to update ECU software over the air; this is already done in some cars, but as far as I know current cars allow an engine ECU update only with a physically present tester. A correctly implemented over-the-air-updates can be made as secure as updates that are deployed by the car dealer with physical access to the car. In practice, however, the physical access makes it harder to exploit vulnerabilities that may exist, thereby limiting the severity of the issue.

I'm not aware that proximity to test labs or testing time frames has been used; if such a mechanism would be employed, and would be detected, it would very clearly show the intention to cheat.

11. What do you consider are the preconditions for accurate type-approval testing by the technical services? In connection to this, do you believe that EU legislation should give type approval authorities and technical services the mandate to undertake reversed engineering and access the software? I refer to the fact you mentioned in your answer to written question 9 for the hearing of Mr Daniel Lange, that there are so many input variables that exhaustive testing of all combination is impossible to do. What is then practically and procedurally necessary to organise? Mr Lange, in his testimony to the EMIS Committee, suggested that: "When a car is type-approved, the code that makes the engine – and all the other safety-related functions – run, needs to be presented to the regulators, including all of the build chain, so that the car can be reprogrammed, basically, without anybody being asked at the OEM, and it will have exactly the same bytes in there as it has when it ran off the factory track". Do you share this view?

There need to be a proof that the software that was delivered in the car matches the description and documentation that was used to pass the type approval testing. As such, I do share the view of Mr. Daniel Lange that the manufacturer needs to supply not just the source code, but also all tools that are necessary to produce the resulting binary that will be installed in the car. Further, it must be possible to verify that the binary installed on a given car matches the original binary that was certified.

Reverse-engineering is not necessary if the sourcecode is available for review, is well-understood (for example, implemented in a well-understood programming language), that there is a way to validate that the software installed in a car matches the source code.

Felix Domke, Lübeck, Germany, September 7th 2016