Increasing resilience by finding unknown vulnerabilities

Aviram Jenik, Beyond Security

More than one million cars were recently recalled by Fiat-Chrysler when a new attack on the Engine Control Unit (ECU) was made public. The attackers had no access to the code; the attack used a pure "black box" approach.
All companies are in a rat race - trying to find vulnerabilities in their networks, and then patching them (or blocking them with defense tools). But some, like the auto manufacturers have bigger problems: vulnerabilities in automotive systems.
This talk will describe technologies that allow organizations to identify as yet unknown vulnerabilities in their own and in 3rd party products, thereby considerably increasing resilience against such attacks; it will explain how a "resilience certification" process can be established to quantify whether a device, an application or an entire car is reasonably secure, in a systematical and repeatable process. It will also discuss what to do once such an unknown vulnerability is discovered.
The main concepts in the talk will be black box testing and fuzzing, with a special focus on automatic testing that requires little security expertise and can be done (once the guidelines are established) by junior or inexperienced personnel, making the process efficient and scalable. Examples will be given for CAN-BUS and OBDII which are unique to the automobile industries, as well as some more common examples used by network devices and Internet servers.

The recent attack on Fiat-Chrysler's "Uconnect®" system made headlines recently. What wasn't emphasized was the fact the attackers did not have any internal knowledge about the system they were hacking. The well-documented attack was a pure "black box" approach, where the attackers try to find weaknesses from the outside. How successful was it? The attackers were able to gain complete control over a Jeep remotely, while a driver was in it, hopelessly trying to operate it against the orchestrated attack. Of course, this is one attack that was made public – we are left to wonder how many other such attacks are kept secret.

Most organizations have an established vulnerability management program where they monitor and fix security holes as those are published. For example, Microsoft releases vulnerabilities on a certain day every month, allowing companies to check if they are vulnerable and fix as needed. Vulnerability intelligence is also at a developed stage where organizations can receive prior information about vulnerabilities that are available in the underground and have not yet been publicly disclosed. There are currently stable methodologies (even if sometimes difficult to follow) to increase resilience against known attacks.

But this kind of program only handles known vulnerabilities. For some organizations, managing known vulnerabilities is not enough since they face a real threat from targeted attacks (aka "Advanced Persistent Threat" APT) that may use zero day vulnerabilities developed especially for this task; to penetrate the defenses. The automotive industry must at this point assume that it is a primary target for dedicated attackers and that fixing vulnerabilities after they have been made public is not going to give their customers much confidence in their products.

In fact, in most of the first world, the threat of the 21st century is a direct cyber-attack. In that scenario, each rival has a secret stash of zero day vulnerabilities that are meant to target some specific country, organization or infrastructure component. In this threat scenario, no one knows about the weaknesses that have been discovered

except the potential attacker. There is no advance intelligence to gather, and no public information to assist. How can a potential target increase resilience against an attack that may come through an unknown vector?

Until now, governments and organizations were completely powerless. They often have not been able to secure copies of the original code of the products used within their borders or products. Vendors are naturally reluctant to surrender their intellectual property. In addition, even if the source code was available, evaluating it for security weaknesses is very difficult and consumes much time and manpower. It‘s hard to give actual success stories of cases where having access to the source code enabled organizations to actually find security holes in 3rd party products. It seems that at some point this was a mere ceremony – the vendor provided the code which was never checked. All the disadvantages with nothing to show for it.

The alternative is to use a 3rd party such as a certification agency following a certification standard (such as “Common Criteria”, ITSEC, COFRAC, and ANSSI) with the hopes that they have the expertise, along with enough access to the product‘s internals, to evaluate it whether it was secure. But this alternative suffered from a fatal flaw: the certification process is long and as the speed of new model development accelerated, certifying each version of every component by every supplier becomes impractical. To compound the difficulty: Although vendors can be forced to cooperate with the certification agencies, once the certification process was finished they no longer had any obligation toward the certification agency, nor did the certification process have any enforcement over future version changes (sometimes drastic) in the product that made the previous certification meaningless.

Without the ability to trust a 3rd party, manufacturers have been helpless in the face of zero day vulnerabilities and targeted attacks. However, the new concept of “Black box Testing” may change this picture completely.

The effectiveness of black box testing

How well does “black box testing” work? Popular vulnerability databases such as “SecuriTeam” and “OSVDB” document about 30-50 new vulnerabilities a week. More than 90% of those vulnerabilities are discovered using black box testing techniques. In fact, the recent vulnerabilities that made it to the headlines - “Heartbleed” is the outstanding example in recent memory – were all found using black box testing techniques, even in cases where the source code was available. “Heartbleed” uncovered a vulnerability in OpenSSL, which is an open source project. The vulnerability had been present in the code for many years and was not found by manual source code audits despite being used by thousands of applications including by major security vendors. A black box test picked up this vulnerability that was previously hidden for decades demonstrating the superiority of this technique.

A recent wave of attacks on closed systems – routers, industrial control systems and as mentioned – car ECUs, demonstrates how effective this attack is against closed systems. We do not yet have documented examples of attacks on critical infrastructure such as water, gas and train systems, but clearly the same technique would work for them as well.

In addition to being a powerful technique for finding issues, ‚black box testing‘ has an additional strong characteristic: it does not require the source code. In fact, by definition, the test is done on a “black box”. We need to know almost nothing about the product or device we are testing, other than the protocol it is using to communicate.

Operational efficiency

The automated nature of black box testing and especially the specific methodology of ‚fuzzing‘ enables repeatable testing in a lab environment with very little manual intervention. In particular, “exhaustive fuzzing”, which will be explained in the talk, provides a process to deliver the widest range of test cases with very little manual work – letting a machine do the grunt work.

The result is the exposure of flaws that will be open to attackers after product release. Since the tool needs to only be set up once, it is possible to repeat these tests for new versions, or for similar systems. For example, a certification agency can set up a lab where certain types of devices are tested by setting them up in the testing environment and then testing against a pre-defined benchmark that is only configured once. In several days millions of attack scenarios can be tried out automatically, giving a wide test coverage.

The main characteristics of black box testing

The first thing to understand about black box testing is that common or known attack signatures are not employed. This is unlike other dynamic testing tools and Vulnerability Assessment tools such as Vulnerability Scanners or Vulnerability Management systems. The idea behind black box testing is to find unknown vulnerabilities; by definition, those cannot be found using already known attack signatures.

In addition, black box testing is applicable to all protocols. For example, the current focus is on CAN-BUS and OBDII; but while these protocols are important, they may not be the only input to programs used in automotive systems. Other protocols such as Bluetooth and Wi-Fi must be tested and can be thoroughly investigated using the same testing tool. The more distant future seems to be heading towards file-based attacks.

File-based attack is especially dangerous since it can “jump over” an air gap. Imagine, for example, an automotive network that accepts files by USB, or by update processes. Any file, such as PDF or JPG image, perhaps a configuration file or an anti-virus update can carry a dangerous payload in them. Any automotive program that accepts a file type will be vulnerable. As a simple example, Acrobat reader, the most popular PDF reader, suffered over the years from a multitude of vulnerabilities that allows someone to create a malicious PDF file that would essentially carry a payload that will be executed on the target machine.

This would be true even if the file was scanned by an anti-virus or a firewall and even if the PDF file was manually transferred into an isolated network, like an automobile – essentially jumping over the air gap. Such a vulnerability can completely compromise the isolated network and can only be found using a black box testing process since it may be a vulnerability that is unknown at the time of product release.

Many devices today, from simple music players to automotive systems process mp3, wav and jpg files and could all be vulnerable to such an attack vector.

Testing proprietary protocols

Black box testing shouldn‘t just be limited to automotive components that use common protocols, either. The most dangerous vulnerabilities exist in products and applications that use proprietary protocols that have not been subjected to rigorous testing. So black box testing must support proprietary and custom protocols as well. In fact, when put into practical use, we find that proprietary protocols are the weakest protocols. The writers of these protocols believe that no one can attack the protocol since its format is unknown. They are very wrong: it‘s possible to attack (as demonstrated by the black box test approach) and when attacked, weaknesses are immediately exposed.

The black box testing process also has two additional and opposite (but complimentary) features: It enables us to cover many different potential attacks (a large attack search space) while allowing us to prioritize which attacks will happen first. This enables us to choose between a quick, and less thorough check and a longer, much more in-depth test. A common use for it would be for organizations to test some products (to be deployed in a less sensitive networks) for a few hours or perhaps a day, while subjecting other products to a more complete, very rigorous testing to potentially uncover more weaknesses over a course of days and weeks.

Another feature, not a must but very much a ‚nice to have' is the possibility to include black box testing in a protocol compliance process. Since we are interested in security weaknesses of any kind, it would be good to be able to test across the entire protocol. This would reveal vulnerabilities in obscure parts of the protocol – documented features that are rarely used. A glaring example is a vulnerability in the PDF file specification, found more than a decade after publishing the specification, which basically allowed any attacker to embed commands that would be executed by the PDF reader, on the target machine. This direct attack was hiding in plain sight, and went unnoticeable until someone decided to go over the entire protocol with a black box fuzzer to look for weaknesses.

Fixing problems to increase resilience

To be clear, our purpose is not to find problems but to help fix them. However, there is a surprising consequence to black box testing: since the weaknesses found are very specific and describe the attack in detail (in a way that enables re-creation) it becomes very easy to fix it.

If the test is being done by the actual developers, the development team can see how to attack works and fix the application to prevent the attack (after all, the attack uses a weakness that should not be there).

If the test is done by a 3rd party with no way to modify the application, it is often possible to block the attack by filtering or masking the attack based on its parameters. Firewalls, IDS or filtering software can assist in preventing such attack.

That may be one of the most surprising aspects of black box testing: Fiat-Chrysler had to recall more than 1 Million cars at a huge expense, where it instead could have found the weakness while the software was being developed and fix the problem at almost zero cost. By waiting until the product is released, the cost of repairing that one flaw sky rocketed.

Real life examples

The author of this paper, while doing a black box test on a very popular SMTP server, stumbled upon a documented, public, feature in the SMTP protocol that allowed attackers to easily bypass content filtering software, including firewalls, anti-viruses and IDS's. This feature was so out in the open that it was implemented in popular software like Outlook and Outlook Express (as well as most of the alternatives), with the entire industry being unaware of the huge hole this feature exposed. This attack was went unnoticed for decades, and only a black box fuzzing test by an objective and unbiased machine found it. Covering the entire protocol would protect against such hidden issues.

Additional points to consider

Black box testing by definition tries to cover a huge attack surface. Prioritization is one way to help cover that space efficiently, but another way is by devising a scalable test process. The black box testing process needs to be scalable for use with multiple processors and/or multiple machines to speed up the process. In an age where processing power is cheap, the ability to trade computing power for a faster test is critical.

When doing black box testing we must make sure not to fall into the "false positive" trap. Just like the story about the boy who cried wolf, if we identify vulnerabilities that end up being false, we risk de-sensitizing the organization and thus when a real critical vulnerability is found, no one will listen. The result of the black box testing must be completely accurate with tests and the resulting failure easily duplicated, while being sensitive enough to even the slightest attacks such as off-by-one attacks.

Can all of these thing be done? The author of this paper thinks the answer is yes. The talk outlined here will present a method of black box testing (a method called "exhaustive fuzzing") which can be used to uncover unknown vulnerabilities practically, does not require the source code, can be used

to cover the entire protocol but also has built-in prioritizations. Exhaustive fuzzing is scalable enough to allow reducing the testing time drastically by employing more computing power, and due to its exhaustive nature it can work well for compliance and certification testing. Exhaustive fuzzing is extremely accurate, and will only report vulnerabilities when there is a high chance that they are in fact exploitable. To cap it off, exhaustive fuzzing can be used for any protocol, over any medium, whether network, wireless, file-based attacks, memory attacks and direct CPU attacks.

If this sounds too good to be true, let's double the ante by adding the fact that exhaustive fuzzing can be done by a relatively untrained personnel and does not require much expertise after the initial threat modeling and configuration is done. It is therefore possible for a security expert to build the model for certain protocols as used in automotive applications (for example, CAN-bus or OBDII) and let less experienced team members run the tests themselves.

After demonstrating "exhaustive fuzzing" for known and documented protocols, and if the time permits, we will show how exhaustive fuzzing can be done for unknown or proprietary protocols.

In conclusion

The talk will describe a practical way to enable organizations test 3rd party products and applications for unknown vulnerabilities. This, without needing to have access to the source code and with no vendor cooperation. In fact, nothing is needed besides a running product and basic knowledge about the protocol (which can be either well defined, or obscure or proprietary). All this, with high accuracy.

And by employing very smart tools that can be used by someone who is not necessarily a security expert.

Aviram Jenik
Beyond Security
19925 Stevens Creek Blvd.
US-95014 Cupertino, CA
Tel. +1-800-801-2821
aviram@beyondsecurity.com
www.beyondsecurity.com