# Network Security for Vehicles on the Internet-Of-Things(IOT). Part 1 Over-The-Air(OTA) Updates

As their complexity has increased, along with their degree of connectedness automobiles have taken an increasingly prominent role in the internet-of-things(IOT). Consumers have come to expect accurate navigation, streaming entertainment and timely software updates for their cars as well as their smartphones.

With this expanded functionality and connectivity has come the vulnerabilities of any embedded system connected to the global network. In addition to safety and reliability, it is critical for the designers of automotive embedded systems to build in the security which will allow them to remain reliable, safe and connected.

In the world of "Stuxnet", "Duqu" and other demonstrated embedded systems vulnerabilities(see Jeep Cherokees), it is essential that vehicles incorporate multiple overlapping layers of security. Particularly as cars achieve greater levels of autonomy, concerns for public safety will require that they adhere to standards comparable to those of critical infrastructure.

Any connected embedded system must assume that it can and will be scanned for vulnerabilities. A new connected device can be expected to be remotely probed within the first 24 hours, and possibly within the first 45 minutes after joining a network.

## A Series on Security

In this series we'll examine network security as it applies to vehicles connected in the IOT. We'll examine threats, countermeasures and best practices for designing secure embedded automotive systems.

Specific topics we'll examine:

- Over-The-Air(OTA) Updates
- Designing a Secure System Architecture - this will include *secure-boot*, principles of *software-design* & *hierarchies of privilege*
- Intrusion Detection and Anonymization

## Potential Attackers

An embedded system may be probed or attacked by a variety of agents with varying degrees of skill, resources and determination.

Potential attackers include:

1. Curious individuals with limited skills and resources - potential hackers simply seeing what vulnerabilities can be exploited. (That individuals should not be underestimated was clearly demonstrated with the tram system in Lodz.)
2. Criminals Organizations. These agents are interested in monetary gain through compromised financial credentials or extortion.
3. Competitors. Motives could range from the passive theft of trade secrets and technology to sabotage. Those attacks could employ significant skills and resources.
4. State actors. Such entities may have early unlimited skills and resources and potentially be in possession of multiple zero-day exploits.

Designing systems to address threats from potential attackers should therefore be comprehensive and not assume limitations in skill or resources.

### An Attacker's Level of Knowledge of a Targeted System

A system's potential vulnerability is in part determined by the level of understanding an attacker has of that system.  The three general categories of attack sophistication are:

1. Black Box -  little or no knowledge of the system's type or internal structure. This sort of attack would be a naive probing of a newly connect system in hopes of leveraging common system exploits
2. Grey Box - some information about the target system is known, potentially including: the type of system (vehicle vs. other embedded system), OS type and version and/or hardware chipsets. This sort of knowledge might be gained by passive observation of traffic (packet-sniffing), casual probing, online research and general inference.
3. White Box - all system information, including hardware specifications and complete sourcecode is available to the attacker. This level of knowledge could be the result of meticulous study & reverse engineering or insider information. Any potential vulnerabilities are assumed to be known by the attacker.

Ideally, attacks would be constrained to the first, "black-box" category, and one facet of security is attempting to limit the amount of system information an attacker is able to collect. Unfortunately, the most conservative security posture assumes that no vulnerability is too obscure to be exploited. The best and safest policy is to assume attacks will be "white-box" in level of sophistication.

## Secure Over-The-Air(OTA) Updates

The ability to quickly respond to newly discovered security exploits (zero-day hacks) and address safety and performance defects with software patches are essential for automotive

systems.  However, the remote updating of firmware creates additional vulnerability by creating channels for the introduction of malware.
Additionally, the act of transmitting sourcecode updates over a communication channel potentially increases the likelihood of more sophisticated “grey-box” and “white-box” attacks.

Whether the mechanism is a disconnected update via thumbnail drive, or OTA, the design of the update process should take the following into consideration:

1. Host Authentication - an automotive system should be able to verify that an incoming software update originated with the OEM or some other sufficiently credible entity. If the source of an update cannot be verified, the update could come from a malicious source and include malware.  It is also desirable for requests for updates from remote embedded systems be verified as originating from legitimate sources. If an attacker is able to spoof the identity of an embedded automotive system and request updates, it greatly increases the volume of data that can be analyzed for vulnerabilities. Even if updates are encrypted, the ability to gather unlimited amounts of ciphertext make cryptanalysis far easier for a potential attacker.
2. Encrypted Channels - if the content of software update messages are not securely encrypted, the embedded code for the system is being shared publicly and is available for analysis by anyone. Access to the source-code for a system makes finding flaws and vulnerabilities far easier, thus turning a black-box into a gray or even white-box.
3. Integrity verification - if an attacker is able to intercept legitimate updates and modify the contents of these messages without detection, that attacker could insert malicious code. Even if the parties on both ends of the update transmission are verified, it should not be assumed that the content of the update message could not have been altered.

Underlying any capability for securely distributing updates is encryption. The two mechanisms available are:

- Symmetric Key Cryptography wherein two parties securely exchange data over a channel using a single shared secret. A shared key, which must be known to both parties to be used to both encrypt and decrypt messages. This has been the only form of cryptography available through most of history. It suffers from the drawback of being entirely dependent on the shared key remaining secret. Once the key is known by an attacker, the code is “cracked” and all data that has been communicated over the channel becomes readable.
- Public Key Cryptography (invented in the 1990s) is based on sets of public-private key pairs. The public key can be distributed and used to encrypt messages that may only be decrypted using the private key also, the public key can be used to verify the origin and integrity of messages signed using the private key.

These mechanisms are often used in tandem wherein public keys are used to exchange temporary symmetric keys which are then used to encrypt the channel.

The applicability of public-key encryption for system updates is fairly straightforward. If a public key of a pair can be shipped with every embedded system, and the private key held by the manufacturer, any system update can be signed using the private key and verified by the embedded system prior to applying the update.  Since both the source and content can be verified, the update can be guaranteed to be free of malware.