Hot fuzz

By Sami Petäjäsoja, Jarkko Lämsä and Anna-Maija Juuso

New access technologies make 4G, the new generation of wireless networks, more vulnerable but also more testable

Sami Petäjäsoja, **Jarkko Lämsä** and **Anna-Maija Juuso** explain fuzzing

Long Term Evolution (LTE) is the emerging network standard for wireless communications. Networks based on it are also referred to as "4G".

LTE differs from earlier standards in that it is completely IP-based. This will improve capacity, speed and profits, but it brings Internet security risks into telecom networks which until now have been closed. The new and more powerful access technologies it enables make LTE networks more vulnerable.

One approach to security is to try to stop malicious data entering the network. However that adds to complexity, expanding the attack surface of the network system. As long as software defects exist, so do potential exploits, and adding more software can increase both. It is better therefore to concentrate on detecting defects for removal and one way to do this is to test robustness of software using the underlying protocols. The fact that IP is an open standard makes this relatively easy, leading to novel test automation techniques such as "fuzzing" which can be used to assure reliability and security in applications of emerging technologies including LTE.

Why and where LTE is vulnerable

The first step in assessing security risks is identifying critical interfaces. In mobile telecoms, we must study the network boundaries. For example, when a mobile phone user accesses different carriers while travelling abroad ("roaming"), core functions are performed by components in those carriers' networks. In order to charge the user, carriers must allow the roaming network provider access to their subscriber data, creating a trust boundary. This requires a high degree of robustness from the network components involved. Also, LTE is backwards compatible, supporting 2G and 3G access to enable smooth transition from earlier technologies but exposing networks to security threats. Previous technologies contain legacy features such as Wireless Application Protocol (WAP) which are dangerous to LTE because they are rarely used and require their own safety features, adding complexity.

The most critical interfaces in the LTE network are the interface between the off-site access point base station (eNB) and the core network, and the interface between the core network and the Internet (see figure 1). These are the two interfaces carriers can least control. High-level protocols, in particular IPv4 and IPv6, pose the biggest risks as they are the most open protocol layer used and the easiest one to attack due to the large number of ready tools.

In 2G/3G networks all traffic passes through a radio interface and is controlled. Unlicensed Mobile Access (UMA) enables handsets equipped with Wi-Fi or Bluetooth to access GSM or GPRS core networks through unlicensed public and private

wireless networks, but UMA traffic is still controlled by the UMA Network Controller (UNC).

LTE however provides for the use of femtocells, which are like home routers and connect in-home cellular access points via a home broadband connection to a carrier's switching system network to establish a localized cellphone service. Femtocells can access the carriers' core networks directly via the internet (figure 2).

Earlier technologies did not enable users to program the handset radio features, but LTE now makes this possible. For example malicious data packages designed to open within the core network can be created.

Using fuzzing to detect vulnerabilities

In robustness testing invalid, unusual or unexpected inputs are made to the system under test to establish how it behaves, externally and/or internally, when it receives them. If an incident occurs then there is a vulnerability that could also be exposed in production.

Fuzzing is a form of robustness testing which focuses on communication interfaces and the discovery of potential security-impacting issues such as overflows and untrapped runtime errors. Any defect is a potential threat to both security and functionality, because attackers search for vulnerabilities in a similar way, looking for an input that produces an abnormal response. They then refine the input, trying to find a combination that causes their desired behaviour. However, not every failure is caused by illicit activity of this kind. Coincidental combinations of inputs and circumstances can also trigger failure: usage peaks are one common contributor. The most effective way to reduce risk of failure is to improve code quality, and using fuzzing throughout the development process is an easy method of doing so.

The strength of fuzzing is its unparalleled ability to find unexpected vulnerabilities. It can be used at all test levels, including on individual software components, enabling defects to be detected and repaired and so software to be retested and regression tested as early as possible. Fuzzing is a very representative test technique because it targets the problems attackers would find more systematically and far more thoroughly than manual exploratory testing. It uses automation to create and execute thousands or even millions of misuse cases for every use case.

Mutation-based and model-based fuzzing

Mutation-based fuzzers are created by capturing network traffic, breaking down the structure of the message exchanges within it, and tagging each element with metadata, which can then be used to automate their recombination to form new structures. The weakness in this approach is that the traffic captured is only a sample; how representative it is is unknown and it tends not to include patterns caused by rarely-used legacy technologies and features such as WAP. Mutation-based fuzzing is often used for very new or obscure protocols and other technologies which are not well specified.

Good specifications allow the use of model-based fuzzing. The specification is used to identify the data elements and also provides other protocol- or format-specific information, for example the boundary values of each element. Model-based is more systematic than mutation-based test generation: the specification allows the protocol to be understood rather than just sampled. Thus the model-based approach produces fewer test cases, because the specific information enables redundant or impossible test cases to be eliminated. The resulting shorter text execution times allow tests with high defect-finding potential to be run more frequently during development, for example on every build, shortening the time between defect introduction and detection and therefore reducing the cost of repair and retest ■

Sami Petäjäsoja is senior product manager, Jarkko Lämsä is a robustness and security testing specialist, and Anna-Maija Juuso is a communication and marketing specialist at Codenomicon, which produces a range of protocol-level security testing tools.

Figure 1: LTE network boundaries

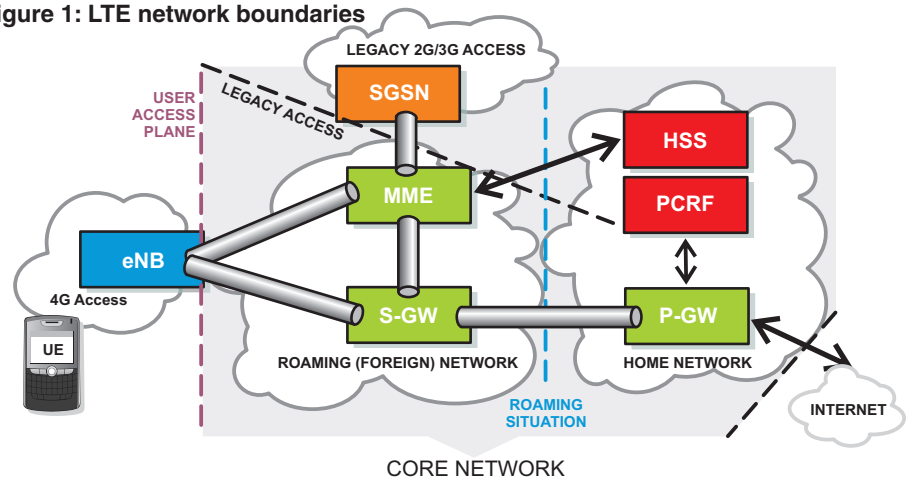

Figure 2: access networks