

## Validating the security of MIDP implementations

**Gemplus' Java Security Technologies Department provides an innovative security evaluation service for J2ME MIDP 2.0 mobile handset implementations using a pioneering in-house midlet Test Suite composed of over 300 unitary tests.**

Recent security evaluations by the Java Security team undertaken on range of commercially available J2ME handsets revealed various fundamental security flaws, mainly due to an inadequate interpretation and implementation of Java security guidelines for the handset's KVM. These security flaws open the way to an array of potential risks that can impact both the subscriber and the mobile network operator and which can lead to denial of service, theft of confidential subscriber data and even to illegal access to handset manufacturer's IP through the operating system.

Used as an essential tool within a rigorous evaluation process, the MIDP 2.0 Test Suite will enable a security risk analysis of MIDP, J2ME API and proprietary API implementation.

### Evaluation methodology

The security evaluation service is based on a three-step proven methodology.

- **Evaluation** of unitary strengths and weaknesses of the MIDP 2.0 & API implementation using the midlet Test Suite.
- **Qualification of the security flaw** with relevant criteria.
- **Analysis of the risk** through use case scenarios and recommendations to minimize risk when feasible

### Test Suite core module description

Test midlets have been developed by the Java Security team to cover each of the following domains:

**Downloading mechanisms.** MIDP 2.0 introduces the notion of trusted/untrusted applets which allows to define restrictions to resources for each midlet. To obtain privileged access, a midlet must be assigned to specific pre-defined domains, and properly signed. In order to be downloaded, installed and granted associated permissions, the device must successfully verify the signature.

The midlet tests in this category involve testing this process and trying to bypass the permissions to access unauthorized resources.

### Sand-boxing and Security

**Manager mechanism.** By default MIDP applications are not trusted, and are assigned to untrusted domains that prevent access to any privileged functionality. These untrusted Java midlets are executed inside a "sandbox" which role is to limit the damage that a Trojan horse can cause if downloaded inside a midlet. Trusted midlets are also limited by permissions and cannot access to ungranted APIs. These mechanisms are tested in order to discover potential limitations and flaws, for trusted as well as untrusted midlets.

**Verification mechanism.** All Java security mechanisms, and particularly the sandbox concept, implicitly require a midlet verifier as a foundation. Without verification no security policy can be enforced. Consequently, the tests involve investigating this verification mechanism in order to pass it. If this is achieved, further tests using unverifiable midlets will be put into place with the aim of breaking through the sandbox and directly accessing the underlying system resources.

**Native methods.** In J2ME, the JVM supporting CLDC does not implement the Java Native Interface and as such it provides only limited security mechanisms. This part of the test suite verifies that the sandbox security model prevents the adding of native functions at application level.

**Buffer overflow.** The tests here involve identifying the methods defined in the CLDC, MIDP 2.0 and proprietary APIs, which are able to manipulate the buffers. The aim of this test section is to read or modify confidential data on the handset by performing an access outside the usual bounds of the buffer.

**Stack overflow.** These tests verify if the stack overflow is systematically detected and if not an evaluation of the possible security-related consequences.

**Threads.** The main objective of these tests is to disrupt thread priority in order to lead to denial of service.

**RMS.** This part of the test suite is concerned with non-authorized access to record, memory saturation and record creation / deletions. MIDP 2.0 APIs allow for the explicit sharing of record stores if the midlet creating the Record Store chooses to give such permission. The test midlets aim to gain a non-authorized access to test this permission process.

**Timer.** Here the tests involve attacks on the Timer command to invoke denial of service.

**Expanded connectivity.** MIDP 2.0 adds support for leading connectivity standards beyond HTTP, such as HTTPS, datagram, sockets, server sockets, and serial port communication, including the possibility to initiate a voice call. For all these connections, the platform asks the user to explicitly acknowledge each request before the action is taken. The Test Suite midlets will try to bypass this mechanism and make a connection without the knowledge of the user.

**Push architecture.** MIDP 2.0 includes a server push model whereby midlets can be registered to be activated when a device receives information from a server. We will test this push architecture that enables to leverage the event-driven capabilities of devices.

For further information contact:  
[gemservicesconsulting@gemplus.com](mailto:gemservicesconsulting@gemplus.com)