M

Mobile Code: Examples, Security Risks, and Controls

Mobile code is code transferred from another system and executed locally, often without a traditional installation step. Examples include scripts and macros.

PUBLISHER
Gridinsoft Support

UPDATED
Oct 8, 2026

READ TIME
3 min

CONTENTS

In this guide

01 What mobile code means
02 Examples
03 What determines the risk?
04 Security controls
05 Warning signs
06 What to do after suspicious execution
07 Source

Quick answer: Mobile code is software or instructions transferred from another system and executed on the local system, often without a traditional installation step. Web scripts, document macros, and other active content are examples. Mobile code is not

inherently malicious, but its privileges, source, and execution environment determine the risk.

01 What mobile code means

In cybersecurity, “mobile” describes code that moves between systems, not code written only for phones. NIST defines it as programs or parts of programs obtained from remote systems, transmitted across a network, and executed locally without explicit installation or execution by the recipient. Physical media can also carry it.

Active content is closely related: a document or page includes scripts, macros, or portable instructions that execute when the content is rendered or an action is taken. Modern web applications depend heavily on controlled active code, so a policy must distinguish required functionality from unnecessary execution.

02 Examples

- JavaScript downloaded with a web page and executed within the browser.
- Office macros or scripts embedded in documents and templates.
- Signed or unsigned browser extensions and add-ons.
- Scripts delivered through email, management systems, notebooks, or automation platforms.
- Portable bytecode, applets, or controls in legacy environments.

Some older technologies named in historical guidance, such as browser Java applets and ActiveX, are no longer common on the modern public web. The security principles still apply to today’s scripts, extensions, document automation, and cloud-delivered code.

03 What determines the risk?

Important factors are the code’s capabilities, the strength of the sandbox or policy enforcement, the origin and integrity of the code, the privileges of the user or host application, and which data or network resources are reachable. A browser script normally operates inside a constrained

origin model, while a legacy control or enabled document macro may receive far broader local access.

A digital signature can authenticate a publisher and reveal modification, but it does not prove the code is safe or appropriate. Certificates can be stolen, publishers can be compromised, and correctly signed code can still contain vulnerabilities.

04 Security controls

1. Inventory which mobile-code technologies and business workflows are required. Remove unsupported runtimes and extensions.
2. Use browser isolation, site permissions, sandboxing, content security policy, and extension allow-lists where appropriate.
3. Block macros from internet-originated documents and permit signed or approved automation only for defined users and locations.
4. Run applications with least privilege. Separate administrative browsing and email from privileged management.
5. Filter, scan, sandbox, or reconstruct untrusted attachments; inspect nested and password-protected content through an approved workflow.
6. Log script execution, parent processes, origin, signature, permission changes, and network behavior centrally.

05 Warning signs

Investigate a document asking to enable macros, a browser page asking the user to paste commands, an unexpected extension install, a script started by an email or office application, or code that attempts to escape its normal sandbox. Unexpected outbound connections and credential or file access after content opens are higher-confidence evidence than the presence of JavaScript alone.

06 What to do after suspicious execution

Disconnect the device if active compromise is suspected. Preserve the original URL, document, email, script, process tree, and network evidence. Determine the privileges and accessible accounts, revoke exposed sessions from a clean device, and hunt for persistence or secondary payloads. Rebuild systems whose integrity cannot be confirmed.

07 Source

The definition and risk model are based on the NIST mobile-code glossary and NIST SP 800-28 Version 2.

NEED MORE HELP?

We are here when the guide is not enough.

Visit the Gridinsoft Support Center for current product guidance or submit a support request with the details of your case.

Open this guide online at support.gridinsoft.com