M

Malicious Code: Types, Detection, Safe Analysis, and Response

Malicious code is intentionally designed or altered to cause an unauthorized effect. It can appear in executables, scripts, documents, extensions, dependencies, or firmware.

PUBLISHER
Gridinsoft Support

UPDATED
Oct 8, 2026

READ TIME
4 min

Malicious code is software, a script, or embedded instructions intentionally designed or modified to create an unauthorized effect. It may steal data, alter transactions, disrupt services, spy on users, maintain access, or weaken security controls.

The term describes intent and behavior, not one file type. Malicious logic can appear in executables, documents and macros, browser extensions, packages and dependencies, mobile apps, web pages, boot components, firmware, automation, or otherwise legitimate software changed in a supply-chain compromise.

Malicious code, malware, exploit, and vulnerability

- **Malware** is malicious software such as a trojan, ransomware, worm, spyware, or wiper. The terms overlap heavily.

- **Vulnerability** is a weakness. It can exist without being intentionally malicious.
- **Exploit** is a technique or code that uses a weakness to produce an unintended result. Authorized researchers may use exploit code in controlled testing.
- **Potentially unwanted software** may be intrusive or deceptive without every instance meeting a malware definition.

A script is not malicious because it is obfuscated, unsigned, administrative, or detected by one heuristic. Context, provenance, requested privileges, behavior, and authorization determine the risk.

How malicious code reaches a system

- phishing attachments, links, archives, and fake document prompts;
- cracked software, repackaged installers, fake updates, and malicious advertisements;
- exploitation of internet-facing applications, browsers, plugins, or devices;
- compromised dependencies, build systems, update channels, repositories, or vendor accounts;
- stolen credentials used to run scripts or deploy tools through legitimate administration systems;
- removable media, shared folders, macros, extensions, and copied commands.

Warning signs and useful evidence

- an endpoint alert with a file, process, script, memory, or persistence path;
- an unexpected parent-child process, scheduled task, service, startup item, module, or browser extension;
- security exclusions, logging, firewall, or update settings changed without approval;
- connections, downloads, account use, encryption, or data access inconsistent with the application role;

- a signed or trusted component executing code from an unusual user-writable location.

Slowness, popups, or high CPU alone cannot identify malicious code. Preserve the source URL or message, filename, hash, signature, timestamps, process tree, command line, network destinations, user context, and security alert before deleting the item.

Safe analysis

1. Do not execute an unknown sample on a normal workstation or upload confidential files to a public scanner.
2. Perform static inspection first in an isolated, authorized environment and record hashes and signatures.
3. For source code, review recent changes, dependency locks, build scripts, install hooks, secrets access, and publisher ownership.
4. For documents, keep active content disabled and inspect macros, relationships, embedded files, and external links.
5. If dynamic analysis is needed, use a disposable sandbox with controlled networking, monitoring, and no production credentials.

A clean static result is not a guarantee. Code may retrieve a later payload, wait for a condition, abuse a legitimate tool, or execute only in memory.

Response and recovery

1. Isolate a confirmed affected system and preserve volatile and stored evidence according to the response plan.
2. Contain the account, delivery channel, destination, or deployment mechanism without destroying the wider timeline.
3. Search for related hashes, behaviors, identities, tasks, packages, and network indicators across other systems.

4. Remove persistence and secondary payloads, patch the entry path, and rotate exposed credentials from a clean device.
5. Rebuild from trusted media when privileged execution, system modification, or scope cannot be established confidently.

Prevention

- patch exposed software and remove unsupported applications and services;
- use least privilege, application control, protected endpoint telemetry, and phishing-resistant MFA;
- restrict internet-delivered macros and unneeded script interpreters;
- verify publishers, dependencies, build provenance, and software update channels;
- maintain offline or immutable backups and test clean restoration.

Reference: NIST: Malicious code.

Malicious code FAQ

Can a legitimate tool be involved?

Yes. Attackers frequently abuse interpreters, remote administration, cloud storage, and system utilities. Investigate authorization and behavior.

Is deleting the detected file enough?

Not when it executed or installed other components. Determine delivery, persistence, accounts, and follow-on activity.

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