CERIAS

the center for education and research in information assurance and security

Software Properties and Behaviors

Pascal Meunier, Purdue University CERIAS

Abstract

Software has moved beyond the encoding of algorithms, to enforcing moral, ethical and legal values, implementing tactics, strategies and essentially the will of designers, coders and organizational (e.g., corporate) entities, or even laws. Buyers, users and communities incur risk due to the deployment of foreign or inappropriate behaviors. Due to code complexity, obfuscation and emergent behaviors, I posit that the systematic study of software behaviors is an important and sometimes the main source of reproducible and objective information on what an artifact (including infrastructure) will and will not do. I contribute definitions of some desirable software properties useful in the context of studying the risks posed by software behaviors: software transparency, purity, obedience and loyalty.

Discussion

Discussing the "behavior" of software artifacts and the infrastructure it depends upon and giving it properties normally associated with sentient beings is not an attempt at superstitious or unscientific anthropomorphism. Rather, it is the recognition that software allows authors to encode complex decisions and policies. Software can force users to register, to choose good passwords or agree to EULAs (End User License Agreements). Malicious software can fool users, and regular software can be fooled by sophisticated users, resulting in "exploits" and vulnerability announcements. Software can spy on users, phone home, and act in the interests of a vendor or even third parties by design. The tactics and goals encoded in software aren't static due to self-update and command-and-control mechanisms. Software can adapt and improve due to the intelligence provided by authors. Additional or updated strategies and goals can be provided by operators.

Many software programs contain unadvertised functions that upset users when they discover them. These functions are not bugs, but rather operations intended by their designers to be hidden from end-users. The problem is not new -- Trojan horses and Easter Eggs were among the earliest instances -- but it is increasingly common and a source of many risks. I define software transparency as a condition that all functions of software are disclosed to users. Transparency is necessary for proper risk management.

Disclosure doesn't by itself remove objectionable functions. They pose risks while being irrelevant to the software's stated purpose and utility, and are foreign to its advertised nature. Freedom from such functions is a property that needs a name: loyalty, nonperfidiousness, fidelity, and purity come to mind, but none of them seems exactly right. I shall call it purity. ``Pure Software'' can theoretically exist without disclosure, but disclosure would be a strong incentive, as previously discussed by Garfinkel. Purity does not mean free of errors or unchanged since release. It's possible for pure software to contain errors or to be corrupted.

Software transparency, purity, obedience and loyalty (c.f. definitions below) are often valued but not explicitly identified. Beyond the obvious information security risks to users,software lacking these properties also poses business risks in the form of loss of reputation, trust, goodwill, sales, and contracts. It may be that transparency alone is enough for some purposes, and others may also require purity, obedience and loyalty. Loyalty is difficult to secure without transparency, purity and obedience. An explicit requirement of whichever is appropriate would decrease risks.

Examples

-Unauthorized back doors obviously fail all desirable properties

-The Comcast network failed transparency by not disclosing the injection of reset packets; it failed purity because the reset packets were foreign to the stated purpose and advertised nature of the broadband internet connections; it failed obedience and loyalty because it violated the control of users' TCP connections in violation of implicit contractual agreements.

-Secret or hidden MMORPG “watch” software (e.g., “Warden” for Wold of Warcraft) fail at least transparency and arguably more

-DRM software for Major League Baseball videos failed obedience and loyalty by unexpectedly preventing users from watching videos, because the DRM server was decommissioned without warning

Loyalty

The software serves only the interests of the appropriate entities. It can't be subverted to serve the interests of third parties, or inappropriately favor some entities

Transparency

All functions of the software are disclosed to users

Purity

Freedom from functions that are foreign to the software's advertised nature or stated purpose

Obedience

The functions in the software are fully under the control of the appropriate entities

PURDUE UNIVERSITY

CERIAS

Discovery Park e-Enterprise Center