Cybersecurity – Requires Multiple Paradigm-Shifts

Erland Wittkoetter, Ph.D., Aug 29th 2022

Seeing problems or issues slightly different could significantly impact conclusions or motivation for actions. Problems can often be ignored or reevaluated because of how we view something. Unfortunately, some current cybersecurity-related ideas and paradigms are detrimental to improving security. Humans in cybersecurity should focus on determining what requires protection and not on behind the scene activities that are better fully automated and irrelevant for any human to get involved.

In the following list, we **assumed that the mentioned ideas are either wrongly accepted by cybersecurity professionals or wrongly ignored, despite being essential**.

(1) **Do not trust CPU/OS**. Cybersecurity is aware that CPU's/OS's complexity is the reason for most vulnerabilities. But this opinion is not sufficiently accepted in IT designs/architectures. Physical separation of CPUs related to security and regular tasks can make a huge difference. Security/control operations are (usually) rigid/static, while all other (regular) operations are versatile and dynamic. Separating security could be prepared similarly to circuit breakers or fuses in power distribution, i.e., it has little complexity in its on-/off-feature. Its operation can also be protected much easier when security is not commingling within the main RAM. As a result, regular algorithms would have no access to security. A data bus cannot be bypassed, which makes it a near-perfect location for separating and providing security components.

(2) **Regular local code validation.** Once installed, software is often considered safe/secure if it does not appear on a blacklist. Software could be modified or reconfigured covertly without being detected as a malicious act. Instead, we could hashcode all local executables and enrich them with data inferred from statistics, which makes them graylisted, or via voluntarily shared registration data from manufacturers, making these hashcodes whitelisted. Once additional data are cached locally, they can be (regularly) used to detect deviations from known software details for instant reporting.

(3) **Software Developers must be made trustworthy.** Medical doctors, lawyers, and financial advisers already have self-regulating rules providing a minimum level of quality control for the public. The same should apply to software developers and manufacturers. Developers' truthfulness with (independently validatable) safety-relevant product disclosures is associated with reputation. Info shared on their software could help significantly to determine what threats or surprises could be expected; deviation from disclosures would automatically raise suspicion. Deceitful exploits in apps would ruin developer's reputation, while creating vulnerabilities accidentally is normal, harmless, and would be ignored.

(4) **Preventing key cleartext disclosures.** Adversaries determined to steal keys could modify via reverse code engineering key processing CPU/OS software. Therefore, key secrecy should mean the main CPU does not process crypto-keys; only protected CPUs are permitted to process secret keys. Keys appearing in cleartext on the CPU must be considered compromised, and hardware that could reveal protected keys must be flagged unreliable for processing secrets.

(5) **Establishing Multi-Unit-Security.** Device components are not independent of the OS. Therefore having security components inter-guarding each other would currently be useless. According to generally accepted design principles, fewer components are considered better than more, but isolated units are prone to misuse. However, if we have independent Multi-Uni-Security, we could use it to have units watching each other if any among them is or was modified.

(6) **Security Execution/Detection must be automated.** We should distrust provided security if humans are directly involved in any non-high-level or operational aspect of security. Only independent automation guarantees reliable rule execution. We can use proactive and preventative conditions for rule violations or damage detection if we know what to expect. All automation methods must be protected against (covert) modifications, reconfigurations, or updates.

Cybersecurity should have tools for two essential goals: Removing the foundation for nation-states' abilities to wage cyberwar and significantly reducing cybercrime. Unfortunately, cybercrime is often the result of human deception or confusion about victims' inherent (sometimes unavoidable) vulnerabilities, all unrelated to malware.

It is proposed that cybersecurity resources are invested in helping users to understand their vulnerabilities and adapt. Having humans involved in technical (routine) tasks unnecessarily increases the risk of new vulnerabilities. Cybercrime will not vanish because of dishonesty and deception, but there should be no damage because of malware. At a minimum, we must eliminate malware, ransomware (data sabotaging), spyware, and backdoors via retrofittable solutions. If done with hardware, we could trust encryption again and its application in eCommerce, online banking, and logistics.

The first goal of computer security should be to end nation states' ability to conduct cyberwars; this is achievable, even rapidly, using a software solution. We could, e.g., allow only hashcoded, whitelisted apps/scripts in RAM via hypervisor/hooksafe-type solutions, a slightly modified approach used to eliminate rootkits for the last 15 years. The goal should be to stop cyber-warfare through an open-sourced grassroots development project. Later, we can also improve the defense against more capable cyber adversaries with separate hardware.

More info: https://NoGoStar.com

 info@NoGoStar.com