Policy Recommendation

The need for proper vulnerability attribution/ characterisation and handling

Context and Findings

Security flaws, glitches or weaknesses (vulnerabilities) are increasingly being discovered in software systems and services.

Software vulnerabilities can be exploited by potential adversaries allowing them to **compromise sensitive data or computational resources**.

Different types of software vulnerabilities do exist, each of them **requiring a different handling approach for mitigation**.

The multi-component nature of modern software systems and services may introduce additional vulnerabilities, and thus increase the advantage of potential adversaries.

EU funding instruments should **support research towards the development of vulnerability attribution/characterization and patching frameworks.**

EU should fund and support research towards the formation of **guidelines and suggestions that ensure the diffusion of accurate information to the public** in regards to the risks and vulnerabilities found in systems and services employing multiple (diverse) components.

The Problem

Security flaws (vulnerabilities) are increasingly being discovered in software systems and services produced by even well-known software companies. These weaknesses can be exploited by skilled adversaries to compromise sensitive data or computational resources.

The multi-component nature of modern software systems and services has the potential to introduce a variety of additional vulnerabilities that increase the adversaries' advantage of exploiting them.

For example, a common system-based vulnerability, namely buffer overflow[1], occurs when the volume of input data exceeds the capacity of the memory buffer and overwrites adjacent memory locations. Buffer overflow can be mitigated using either stack canaries or safe functions instead of unsafe ones.

As another example consider SQL injection (SQLi)[2], which is a web-security vulnerability that allows an adversary to interfere with the queries that an application makes to its database. Conducting successful SQLi attacks may result in unauthorized access to personal/sensitive information (e.g., credit card numbers, passwords, health records), subvert an application's logic, etc. SQLi attacks can be prevented by using parameterized queries, validating the user's inputs, and enforcing appropriate privileges with strict access rights.

The aforementioned examples are just two instances of a large pool of possible vulnerabilities introduced by various components of different nature.

The Scene

We live in an era where the majority of systems and services are comprised of multiple (different) components. These components often have diverse characteristics, and thus require different handling approaches for mitigating their vulnerabilities and restoring the system's security. Simply put, different bugs require different handling methodologies.

What is important to note is that many of these vulnerabilities lack proper treatment that may result in increased security/privacy risk. This is mainly because the majority of security incidents are often characterized by general terms and descriptions, such as advanced persistent threat (APT), which fail to capture and report the exact root cause(s) that led to those incidents.

For example, consider an authentication system based on face recognition[3] that has been bypassed. The root cause of the bypass can be a buffer overflow[1], social engineering[4] or a model inversion attack[5]. Characterising the vulnerability that led to this security incident using a general (non-specific) term, such as APT, may result in an incomplete mitigation strategy as well as misinforming the public about the actual root cause of the problem.

Thus, while it is possible to handle bugs individually, there is a growing need for a generally applicable framework that accurately characterises/attributes the vulnerabilities of a system composed of multiple components of varying nature.

The Implications

Handling all types of vulnerabilities using the same approach/methodology will most likely result in an incomplete mitigation, and thus increased risk of exploitation. Especially in the case where security-critical and privacy-sensitive systems are exploited, the potential consequences can be severe, even causing fatalities.

Not only that, but a non-comprehensive vulnerability attribution framework may also result in the misinformation of the general public, which in turn might cause negative opinion or security/privacy concerns towards the wrong technological component.

As a result, it is crucial that the handling of different vulnerabilities must be made based on their type. This will result in a comprehensive solution that is devoid of any security, privacy, or ethical concerns.

Policy Recommendations

If EU-supported funding initiatives do not change, we will probably end up with an environment that is hostile to the development and fruition of systems and services employing different types of technological components. This will probably have an adverse impact on the European software ecosystem, the development and fruition of novel inventions and eventually on European digital sovereignty. To reverse this trend, we have three policy recommendations:

1) **Fund and support research towards the development of comprehensive vulnerability attribution and characterization frameworks**
The fact that modern systems and services employ multiple components of different nature makes the detection of their vulnerabilities a highly non-trivial process. Thus, having available defences for up-to-date threats is not useful if we cannot effectively/efficiently (and automatically) determine from which specific vulnerabilities the tested systems suffer from. For this reason, the EU should fund and support research towards the development of comprehensive vulnerability attribution and characterization frameworks that are designed for being applicable to systems and services employing multiple components of different nature. Such frameworks should be easy to deploy by even non-experts in the security field who wish to incorporate different technological components into their applications.

2) **Fund and support research towards the development of frameworks for assisting developers during the vulnerability patching procedure**
Solely detecting the vulnerabilities of modern systems and services comprised of multiple diverse components is likely not enough; we need to mitigate those vulnerabilities and restore those systems' security. In addition, it is worth mentioning that software developers are often non-experts with regard to the security aspect of every single component that they incorporate into their applications. Even if they are experts in certain components, they may not be aware of the potential security and privacy risks that arise when combining those components with other components of different nature. Thus, the EU should fund and support research towards the development of frameworks for assisting software developers into patching the vulnerabilities found in multi-component systems and services with state-of-the-art defences. The offered assistance can take the form of suggestions/guidelines or, where possible, the form of updates: that is, automatically patching the vulnerable system. Again, such frameworks should be easy to deploy by even non-experts in the security field who wish to improve their systems' resilience against state-of-the-art attacks.

3) **Forming guidelines for the diffusion of accurate information regarding the risks and vulnerabilities of modern software systems and services**
The EU should fund and support research towards the formation of guidelines and suggestions for the dissemination of accurate information with regard to security/privacy vulnerabilities found in modern software systems and services. In particular, generic vulnerability attribution might cause misinformation and, as a consequence, the negative public opinion and/or security/privacy concerns towards the wrong technological component. For example, the reason that an authentication system has been bypassed might not be a vulnerability of the algorithm per se, rather than a buffer overflow (or the opposite). Thus, we need to form appropriate guidelines and suggestions that require the diffusion of the exact root cause(s) of each security incident. Doing so will facilitate the dissemination of accurate information to the public and avoid negative opinions and/or loss of people's trust towards recent technological advancements.

References

[1] Larochelle, D., & Evans, D. (2001). Statically detecting likely buffer overflow vulnerabilities. In 10th USENIX Security Symposium (pp. 177–190).

[2] Boyd, S. W., & Keromytis, A. D. (2004, June). SQLrand: Preventing SQL injection attacks. In International conference on applied cryptography and network security (pp. 292-302). Springer, Berlin, Heidelberg.

[3] Bud, A. (2018). Facing the future: The impact of Apple FaceID. Biometric technology today, 2018(1), 5-7

[4] Krombholz, K., Hobel, H., Huber, M., & Weippl, E. (2015). Advanced social engineering attacks. Journal of Information Security and applications, 22, (pp. 113-122).

[5] Fredrikson, M., Jha, S., & Ristenpart, T. (2015, October). Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (pp. 1322-1333).

Contact

This brief was produced by members of the CyberSec4Europe consortium.
Contact person: Elias Athanasopoulos eliasathan@cs.ucy.ac.cy | cybersec4europe.eu

CyberSec4Europe is funded by the European Union under the H2020 Programme Grant Agreement No. 830929. These results reflect only the view of the authors and the Commission is not responsible for any use that may be made of the information it contains.