Automated software vulnerability discovery, exploitation, and patching

Henry Post, hp2376@nyu.edu, NYU-CS-GY-6813

The current nature of most open-source projects exist as a machine-readable, machine-parseable format. Builds are highly standardized and adhere to a common directory, syntax, and language format. Most open-source software is written in only a handful of languages: C, C++, C#, Python, Java, Go, Ruby, PHP.

Github, as a highly discoverable, open, and transparent code hosting solution, has enabled millions of developers to contribute application source code. Larger projects such as Apache Tomcat (webserver), Visual Studio Code (code editor), TensorFlow (ML library), React Native (JS Server-side framework), etc.

Because of the nature of open-source, any software bugs created are also immediately visible to the entire world. These bugs may go unnoticed for years, later to be discovered by a security researcher, black hat hacker, end user, or anyone else who uses the software.

With the advent of highly discover-able and machine-readable open source projects, as well as automated vulnerability scanning, I believe that exists an emerging field of automated vulnerability scanning of open-source, which could lead into automated exploitation of undiscovered open-source vulnerabilities.

We are already seeing software component (or BOM, Bill of Materials) scanning, such as Snyk (http://snyk.io/) and White Source (https://www.whitesourcesoftware.com/) show up. These detect components with already-discovered vulnerabilities (i.e. known CVEs), but not new vulnerabilities.

Additionally, software such as Fortify (https://www.microfocus.com/en-us/cyberres/application-security), WebInspect (https://www.microfocus.com/en-us/cyberres/application-security/webinspect), and SonarQube (https://www.sonarqube.org/) are offered commercially or for free, to perform software composition analysis (SCA). These tools can allow you to discover new vulnerabilities in software without manual testing.

If an SCA tool were to be fully integrated with a framework that allowed for code scanning, exploit generation, exploit mutation, exploit confirmation, and exploit execution, it would make exploiting security vulnerabilities in open-source projects trivial compared to traditional human-powered methods.

To counter the risk of a wide-scale automated software exploitation tool from being used on many open-source projects, perhaps to great effect, a method must be devised for automatically detecting and patching open-source software vulnerabilities, before threat actors are able to auto-detect and auto-exploit software vulnerabilities.

In this paper I plan to survey the current publicly available research in this paper's domain (Automated software vulnerability discovery, exploitation, and patching), and perform analyses of OSS using freely available automated tools to see how easy it is to automatically discover, exploit, and perhaps even patch software vulnerabilities.

References

- Z. Wang, Y. Zhang, Z. Tian, Q. Ruan, T. Liu, H. Wang, Z. Liu, J. Lin, B. Fang, and W. Shi, “Automated Vulnerability Discovery and Exploitation in the Internet of Things,” Sensors, vol. 19, no. 15, p. 3362, Jul. 2019 [Online]. Available: http://dx.doi.org/10.3390/s19153362

- “Vulnerability scanning of IOT devices in Jordan using SHODAN,” IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org/document/8277814.

- “The Coming Era of alphahacking?: A survey of automatic software vulnerability detection, exploitation and patching techniques,” *IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8411838.

- “A machine learning-based approach for automated vulnerability remediation analysis,” *IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9162309.

- “Robot hacking games,” *Center for Security and Emerging Technology*, 05-Oct-2021. [Online]. Available: https://cset.georgetown.edu/publication/robot-hacking-games/.